UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

GRASP E BUSCA TABU APLICADOS A PROBLEMAS DE PROGRAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS

Autor: Moacir Felizardo de França Filho
Orientador: Vinícius Amaral Armentano

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas – UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Área de concentração: Automação

Banca Examinadora:

Vinícius Amaral Armentano	(orientador)
Débora Pretti Ronconi	EP/USP
Marcone Jamilson Freitas Souza	DC/ICEB/UFOP
Cid Carvalho de Souza	DTC/IC/UNICAMP
Akebo Yamakami	DT/FEEC/UNICAMP

Campinas, 26 de outubro de 2007

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

França Filho, Moacir Felizardo de
F844g GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas / Moacir Felizardo de França Filho. --Campinas, SP: [s.n.], 2007.

Orientador: Vinícius Amaral Armentano
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Planejamento da produção 2. Programação heurística. 3. Otimização combinatória. 4. Pesquisa operacional. I. Armentano, Vinícius Amaral. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: GRASP and Tabu Search applied to scheduling problems in parallel machines
Palavras-chave em Inglês: Scheduling, Heuristic programming, Combinatorial optimization, Operational research
Área de concentração: Automação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Débora Pretti Ronconi, Marcone Jamilson Freitas Souza, Cid Carvalho de Souza, Akebo Yamakami
Data da defesa: 26/10/2007
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Moacir Felizardo de França Filho

Data da Defesa: 26 de outubro de 2007

Título da Tese: "GRASP e Busca Tabu Aplicados a Problemas de Programação de Tarefas em Máquinas Paralelas"

Prof. Dr. Vinícius Amaral Armentano (Presidente): Vinícius A. Armentano

Prof. Dr. Marcone Jamilson Freitas Souza:

Profa. Dra. Débora Pretti Ronconi: Débora Pretti Ronconi

Prof. Dr. Cid Carvalho de Souza:

Prof. Dr. Akebo Yamakami:

RESUMO

Este trabalho é dedicado à programação de tarefas em máquinas paralelas. Dois ambientes são considerados. No primeiro, as máquinas são idênticas e o objetivo é a minimização da soma ponderada de custos de atraso. Todas as tarefas estão disponíveis para processamento no início do horizonte de programação e a cada uma são associadas uma data de entrega e uma penalização por atraso específicas. No segundo, as máquinas são não relacionadas e o objetivo é a minimização da soma ponderada de custos de avanço e de atraso. Instantes de liberação, datas de entrega, penalizações por avanço e por atraso são específicos para cada tarefa. Em ambos, as transições entre tarefas requerem tempos de preparação dependentes da sequência de processamento. Os problemas são resolvidos por meio de GRASP e Busca Tabu. Memória de longo prazo é empregada para melhorar o desempenho das duas metaheurísticas. No GRASP, soluções de elite influenciam a fase construtiva. Na Busca Tabu, estratégias de diversificação e de intensificação fazem uso direto das soluções de elite e também de freqüências de residência. Como pós-otimização, nas duas metaheurísticas, realizam-se religações de caminhos entre as soluções de elite.

Palavras-chave: programação de tarefas, máquinas paralelas, tempos de preparação depententes da seqüência, minimização de avanço e de atraso, GRASP, Busca Tabu, religação de caminhos.

ABSTRACT

This work is dedicated to the scheduling of a set of jobs in parallel machines. Two scenarios are considered. In the first one, the machines are identical and the objective is the minimization of the weighted sum of tardiness costs. All jobs are ready for processing at the beginning of the scheduling horizon and to each one is associated a due date and a tardiness penalty. In the second scenario, the machines are non-related and the objective is the minimization of the weighted sum of earliness and tardiness costs. Ready times, due dates, earliness and tardiness penalties are specifics to each job. In both problems, the transitions between jobs require sequence dependent setup times. The problems are solved using GRASP and Tabu Search. Long term memory is applied to improve the performance of the metaheuristics. A set of elite solutions are used to influence the constructive phase in GRASP. In Tabu Search, diversification and intensification strategies make direct use of the elite solutions, as well of residence frequences. Path relinking between the elite solutions is used as a post-optimization approach.

Keywords: scheduling of jobs, parallel machines, sequence dependent setup times, earliness and tardiness, GRASP, Tabu Search, path relinking.

Este trabalho é dedicado a Laura.
Minha alma gêmea, confidente e companheira.

AGRADECIMENTOS

A Deus, pela vida e pelas pessoas amorosas que coloca em meu caminho.

Ao Prof. Vinícius Amaral Armentano. Por sua amizade, confiança, dedicação e muita, muita paciência para me orientar. Nenhuma palavra é suficiente para expressar de modo adequado minha mais profunda e sincera gratidão.

Ao Centro Federal de Educação Tecnológica de Minas Gerais, pela licença e à CAPES, pelo suporte financeiro.

Aos colegas professores Sérgio Ricardo de Souza e Wilson Luiz de Almeida.

Aos amigos do tempo em que vivi em Campinas: Allan Kardec, Eder Lima, Edilson Bueno, Horácio Duarte, Jeanne Dobgenski, Juan, Ivan Lima, Luiz Rodrigues, Maria Marta, Marcone e Vinícius Jacques.

Aos familiares: D. Elza, Adriano, Alexandre, Dirce, Eduardo, Ferdinando, Geralda, Helena, Irene, Jacqueline, Jadir, José, Léa, Leandro, Maria, Montgomery, Maurício e Rita.

À minha mãe Conceição e meu pai Moacir (*in memorian*), exemplos de coragem, de força, de serenidade e de simplicidade.

À minha esposa Laura, por todo seu amor e apoio incondicional.

Não desanimes. Persiste mais um tanto.

Não cultives pessimismo. Centraliza-te no bem a fazer.

Esquece as sugestões do medo destrutivo.

Segue adiante, mesmo varando a sombra dos próprios erros.

Avança, ainda que seja por entre as lágrimas.

Trabalha constantemente. Edifica sempre.

Não consintas que o gelo do desencanto te entorpeça o coração.

Não te impressiones à dificuldade.

Convence-te de que a vitória espiritual é construção para o dia-a-dia.

Não desistas da paciência. Não creias em realização sem esforço.

Silêncio para a injúria. Olvido para o mal.

Perdão às ofensas. Recorda-te de que os agressores são doentes.

Não permitas que os irmãos desequilibrados te destruam o trabalho ou te apaguem a esperança.

Não menosprezes o dever que a consciência te impõe.

Se te enganaste em algum trecho do caminho, reajusta a própria visão e procura o rumo certo.

Não contes vantagens, nem fracassos. Estuda buscando aprender.

Não te voltes contra ninguém.

Não dramatizes provocações ou problemas.

Conserva o hábito da oração para que ele te faça luz na vida íntima.

Resguarda-te em Deus. Persevera no trabalho que Deus te confiou.

Ama sempre, fazendo pelos outros o melhor que puderes realizar.

Age auxiliando. Serve sem apego.

E assim vencerás.

Emmanuel

Adaptação de mensagem psicografada por Francisco Cândido Xavier

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

Capítulo 1

Introdução

1.1 Objetivo

O objetivo do presente trabalho é propor e analisar métodos heurísticos para resolver problemas de programação de tarefas em máquinas paralelas.

Problemas de programação de tarefas estão entre os mais estudados na área de pesquisa operacional, quer pela elevada freqüência com que ocorrem na prática, quer pela sua natureza desafiadora.

Pesquisa realizada por Panwalkar *et al.* (1973) mostrou que respeitar datas de entrega acordadas com clientes era um dos principais critérios observados pelos profissionais responsáveis pela programação da produção. Minimização do tempo total de processamento, dos tempos ou custos totais de preparação e custos de estoque durante o processamento, eram considerados objetivos secundários. Hoje, passados mais de trinta anos da referida pesquisa, o mercado mostra-se cada vez mais competitivo, conferindo importância ainda maior a critérios de desempenho baseados em datas de entrega, mas com uma tônica diferente daquela de tempos atrás, isto é, penalizando, também, a conclusão das tarefas antes do instante em que elas são requeridas (filosofia *just-in-time*). Concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade de disponibilização antecipada de capital, necessidade de espaço para armazenamento e necessidade de outros recursos para manter e gerenciar o estoque.

Ainda de acordo com a pesquisa de Panwalkar *et al.* (1973), 70% dos entrevistados afirmaram que pelo menos 25% das tarefas por eles programadas estariam sujeitas a tempos de preparação dependentes da seqüência. Para outros 13% dos entrevistados, a necessidade

de tempos de preparação dependentes da seqüência chegaria a 100% das tarefas programadas. Em relação a este aspecto, por mais que os processos de fabricação tenham evoluído, não há porque acreditar que a preparação dependente da seqüência de processamento tenha perdido sua relevância.

Apesar da importância prática dos critérios de desempenho baseados em datas de entrega, há poucos trabalhos que os considerem na presença de tempos de preparação dependentes da seqüência.

Este é, pois, o foco dos dois problemas tratados no presente trabalho. No primeiro, o objetivo é a minimização da soma ponderada de custos de atraso de um conjunto de tarefas a serem processadas em um conjunto de máquinas idênticas e continuamente disponíveis. Todas as tarefas estão disponíveis para processamento no início do horizonte de programação e a cada uma é associada uma data de entrega e uma penalização por atraso específicas. No segundo problema, o objetivo é minimizar a soma ponderada de custos de avanço e de atraso. Instantes de liberação, datas de entrega, penalizações por avanço e por atraso são específicos para cada tarefa e as máquinas, também continuamente disponíveis, são não relacionadas. Um procedimento de alocação temporal das tarefas, com inserção de tempos ociosos, é apresentado para o problema com custos de avanço e de atraso.

Os estudos em programação de tarefas dividem-se em duas vertentes: métodos exatos e métodos heurísticos. Métodos exatos podem chegar a soluções ótimas mas são limitados a problemas menos complexos, isto é, com poucas restrições tecnológicas e de menores dimensões. Caso estas não sejam as condições existentes, é necessário recorrer a métodos heurísticos. Estes, por sua vez, não garantem a otimalidade global da melhor solução apresentada, mas, se bem desenvolvidos, podem fornecer soluções de muito boa qualidade e em tempos computacionais aceitáveis.

Do ponto de vista das restrições tecnológicas, os problemas aqui tratados não estão enquadrados entre os mais complexos. Muito pelo contrário, é bastante simples encontrar uma solução factível para eles. Qualquer atribuição de tarefas às máquinas e qualquer seqüência de tarefas em cada máquina pode ser transformada em uma solução factível, bastando para isso que sejam estabelecidas alocações temporais adequadas, que levem em consideração os

instantes de liberação, os tempos de preparação e os tempos de processamento. No entanto, do ponto de vista computacional, são problemas que não podem ser resolvidos em tempo polinomial. É necessário, então, abrir mão da certeza da otimalidade global e recorrer a métodos heurísticos. Esta é a linha do presente trabalho.

Os problemas são resolvidos por meio de GRASP (Feo e Resende, 1995) e de Busca Tabu (Glover, 1986) – meta-heurísticas classificadas como de busca em vizinhança (conjunto de soluções alcançáveis por meio de um determinado tipo de movimento). Duas variantes são formuladas. Uma em termos de movimentos de troca e de inserção, explorados alternadamente, e outra em termos de movimentos *cross* (Taillard *et al.*, 1997) e *Or-Opt* (Or, 1976), também explorados alternadamente. Os resultados de cada variante são comparados para avaliar o desempenho relativo entre os tipos de movimentos.

GRASP é um procedimento de múltiplos reinícios. É caracterizado pela realização de iterações que são compostas pela construção de uma solução inicial, seguida de uma busca local para a obtenção de um ótimo local. A construção da solução inicial é orientada por uma função gulosa. As iterações são repetidas até que seja satisfeito um critério de parada e o melhor ótimo local é selecionado. O desempenho do GRASP básico pode ser melhorado por meio de abordagens que fazem uso de memória de longo prazo. As abordagens investigadas fazem referência a um conjunto de soluções de elite. Na primeira, caracterizada por uma integração entre GRASP e memória adaptativa (Fleurent e Glover, 1999), é feita uma combinação entre a função gulosa usada na fase construtiva e uma função de intensificação, cuja influência é tanto maior, quanto mais freqüente for a ocorrência de um determinado atributo nas soluções de elite, e quanto menores forem os custos das soluções de elite nas quais tais atributos estiverem presentes. Deste modo, as soluções iniciais passam a adquirir características presentes nas soluções do conjunto de elite. Na segunda abordagem, baseada em trabalho de Laguna e Martí (1999), a solução encontrada ao final de uma iteração do GRASP é conectada a uma solução do conjunto de elite, em um procedimento denominado religação de caminhos (Glover, 1996). Este procedimento também é utilizado como estratégia de pós-otimização, ligando as soluções de elite entre si (Resende e Ribeiro, 2005). Outra estratégia utilizada para melhorar o desempenho do GRASP básico é denominada princípio de otimalidade próxima (Glover e Laguna, 1997), que no presente trabalho é definido como buscas locais em soluções parciais.

Busca Tabu é uma estratégia de busca local baseada em memória, essencialmente fundamentada na proibição de certos movimentos. A memória tem estruturas de curto e de longo prazo. A memória de curto prazo tem por objetivo proibir movimentos que levem a soluções visitadas em um passado recente, sendo estabelecida em termos dos atributos que caracterizam a solução. Nos problemas aqui tratados, os atributos são constituídos pelas relações de precedência entre tarefas e também pelas atribuições das tarefas às máquinas (no caso do problema com máquinas não-relacionadas). Dois critérios baseados em relações de precedência, com significados completamente opostos e com diferentes graus de restritividade, são investigados. Investiga-se, também, um critério baseado nas atribuições das tarefas. Experimentos são realizados para definir a melhor faixa de duração tabu para cada critério de proibição. As estratégias de longo prazo dividem-se em estratégias de diversificação, projetadas para levar a busca para regiões pouco exploradas, e de intensificação, por meio das quais procura-se explorar, de modo mais detalhado, as regiões associadas às melhores soluções encontradas até então. São investigadas: i) diversificação por freqüência de residência (Glover e Laguna, 1997), em que as regras de escolha são modificadas a fim de trazer para a solução atributos pouco presentes nas iterações anteriores, ii) religações de caminhos (Glover, 1996), em que a solução atual é conectada à solução de elite mais distante e iii) reinícios a partir de uma solução criada por meio de uma composição que envolve um banco de soluções de elite, de acordo com um procedimento sugerido por Rochat e Taillard (1995), para o problema de roteamento de veículos e que se propõe a integrar as estratégias de diversificação e de intensificação. Como estratégia de pós-otimização, empregam-se religações de caminhos entre as soluções do conjunto de elite.

1.2. Estrutura do trabalho

O trabalho está estruturado em 6 capítulos. O **Capítulo 2** é dedicado ao problema de minimização da soma ponderada de custos de atraso em máquinas paralelas idênticas (problema **PST**). É iniciado pela caracterização do problema e por uma extensa revisão bibliográfica. Descrevem-se uma implementação da meta-heurística GRASP – seus elementos básicos e estratégias para melhorar seu desempenho – e uma implementação de Busca Tabu, com memória de curto e longo prazo, voltadas para o problema em questão. No **Capítulo 3** o problema abordado é o da minimização da soma ponderada de custos de avanço e de atraso em máquinas paralelas não-relacionadas (problema **PSET**). Sua estrutura é semelhante à do Capítulo 2. No **Capítulo 4** são apresentados os resultados dos testes computacionais para o problema **PST**. Primeiramente são apresentadas as variantes desenvolvidas para cada meta-heurística, com a especificação dos valores de cada parâmetro que as caracterizam. Em seguida as variantes são comparadas entre si, destacando-se as melhorias de desempenho possibilitadas e prováveis causas de insucessos. No **Capítulo 5** são apresentados os resultados dos testes computacionais para o problema **PSET**, do mesmo modo como feito para o problema **PST**, no Capítulo 4. No **Capítulo 6** são apresentadas as conclusões e propostas para trabalhos futuros.

Capítulo 2

Minimização da soma ponderada de custos de atraso em máquinas paralelas idênticas

2.1 Caracterização do problema

No problema estudado neste capítulo procura-se definir a atribuição de um conjunto de n tarefas independentes a um conjunto de m máquinas paralelas idênticas continuamente disponíveis, bem como a seqüência de processamento das tarefas em cada máquina e o instante de início de cada tarefa, visando a minimização da soma ponderada de custos de atraso em relação às datas de entrega. Para brevidade em referências futuras este problema é denominado **PST** (das palavras-chave em inglês *Parallel machines*, *Setup times* e *Tardiness*).

Cada tarefa i é caracterizada pelo seguinte conjunto de parâmetros: tempo de processamento, p_i, data de entrega, d_i, e penalização por unidade de tempo de atraso, t_i. Todas as tarefas estão liberadas para processamento no início do horizonte de programação, não sendo admitidas interrupções no processamento de nenhuma delas. A transição entre duas tarefas adjacentes (i e j) demanda um tempo de preparação, s_{ij}, que depende da seqüência de processamento das tarefas. Para o processamento da tarefa i na primeira posição da seqüência é necessário um tempo de preparação inicial, s_{0i}, em que a tarefa 0 é uma tarefa fictícia. Os tempos de preparação respeitam a desigualdade triangular, isto é, $s_{ik} \leq s_{ij} + s_{jk} \quad \forall\, i, j, k$, uma vez que esta é a situação mais comumente encontrada na prática.

O atraso da tarefa i é definido como $T_i = \max(0, C_i - d_i)$, em que C_i é o instante de conclusão do processamento da tarefa i. A função objetivo a ser minimizada é, portanto, $Z = \sum_{i=1}^{n} t_i T_i$.

Na Figura 2.1 apresentam-se a designação e o seqüenciamento de uma solução típica para um problema com 10 tarefas (numeradas de 1 a 10) e 3 máquinas. Cada máquina é implementada como uma lista ligada, iniciada com uma tarefa fictícia (tarefa 0) e terminada também com uma outra tarefa fictícia. De cada tarefa partem dois ponteiros. Um para a tarefa seguinte e outro para a tarefa anterior – este último é particularmente útil no momento de se fazer referência aos tempos de preparação entre as tarefas.

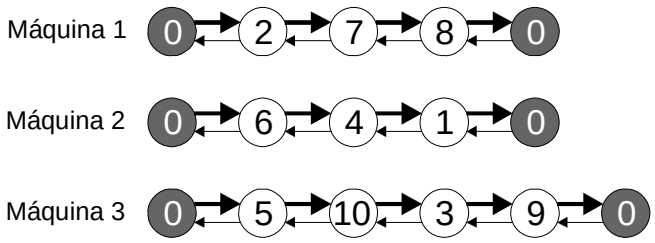

Figura 2.1 – Designação e seqüenciamento em uma solução típica

2.2 Revisão bibliográfica

A minimização do atraso total em uma única máquina, sem tempos de preparação, é *NP-hard* (Du e Leung, 1990). Portanto, o problema aqui tratado também o é, podendo ser resolvido até a otimalidade global, por meio da aplicação de algoritmos ótimos, apenas se a instância considerada for de tamanho bastante reduzido. É necessário, desse modo, o desenvolvimento de procedimentos heurísticos que conduzam a soluções de boa qualidade (baixos custos) em tempos aceitáveis.

Pesquisa realizada por Panwalkar *et al.* (1973) mostrou que respeitar datas de entrega acordadas com clientes era um dos principais critérios a serem observados pela programação da produção. Hoje, passados mais de trinta anos desde a referida pesquisa, o mercado mostra-se cada vez mais competitivo, conferindo importância ainda maior a critérios de desempenho baseados em datas de entrega. Como conseqüência, diversos pesquisadores se viram estimulados a desenvolver métodos exatos ou heurísticos para problemas de programação de tarefas, como mostrado nas revisões realizadas por Baker e Scudder (1990), Abdul-Razaq *et al.* (1990), Cheng e Sin (1990), Koulamas (1994), Kawaguchi e Kyan (1998), Allahverdi *et al.* (1999) e mais recentemente por Sen *et al.* (2003). Nestas revisões, observa-se, contudo, que na grande maioria dos trabalhos não são considerados tempos de preparação dependentes da seqüência, apesar de sua importância para a representação mais fiel dos problemas encontrados na prática. Allahverdi *et al.* (1999) destacam a importância do tratamento explícito de tempos de preparação dependentes da seqüência, especialmente nos casos de gerenciamento de capacidade de produção.

Do ponto de vista dos tempos de preparação, sejam eles dependentes ou independentes da seqüência, os problemas de programação de tarefas podem ser classificados em *batch* ou *non-batch*. Em um problema envolvendo preparação de *batch* as tarefas são agrupadas em *batches* ou famílias. Maiores tempos de preparação são requeridos quando ocorrem transições entre tarefas de diferentes famílias e, em algumas aplicações, preparações mais rápidas, ou de menores custos, são requeridas na transição entre tarefas de uma mesma família (Allahverdi *et al.*, 1999).

Dentre os diversos trabalhos envolvendo máquinas paralelas e tempos de preparação dependentes da seqüência, em um contexto de preparação de *batch*, podem ser citados: Schutten e Leussink (1996), Carlyle *et al.* (2001), Eom *et al.* (2002), S.S. Kim *et al.* (2003), D.W. Kim *et al.* (2003).

Schutten e Leussink (1996) apresentam um procedimento *branch-and-bound* para minimização do máximo *lateness* em um problema com máquinas idênticas, distintos instantes de liberação das tarefas e distintas datas de entrega.

Carlyle *et al.* (2001) consideram o problema de minimização simultânea do *makespan* e do atraso total ponderado para ilustrar a utilização de uma nova medida para comparar algoritmos para problemas que envolvem múltiplos objetivos.

Eom *et al.* (2002) e S.S. Kim *et al.* (2003) tratam da minimização do atraso total ponderado em máquinas paralelas idênticas. Datas de entrega, penalizações por atraso e tempos de processamento são especificados para cada tarefa.

D.W. Kim *et al.* (2003) investigam a programação de tarefas em máquinas paralelas não relacionadas, em que o objetivo é minimizar o atraso total ponderado. Tarefas idênticas ou similares são processadas em *batches*, a fim de diminuir os tempos de preparação e/ou de processamento. Todas as tarefas de um mesmo *batch* têm um mesmo tempo de processamento e uma mesma data de entrega. A penalização por atraso de um *batch* de tarefas é inversamente proporcional à sua data de entrega.

Em termos de preparação *non-batch* em máquinas paralelas, tem-se: Parker *et al.* (1977), Guinet (1991, 1993, 1995), Ovacik e Uzsoy (1995), França *et al.* (1996), Lee e Pinedo (1997), Park *et al.* (2000), Gendreau *et al.* (2001), Kurz e Askin (2001), Weng *et al.* (2001), Hiraishi *et al.* (2002), C. O. Kim e Shin (2003), Fowler *et al.* (2003), Bilge *et al.* (2004), Anglani *et al.* (2005), Rojanasoonthon e Bard (2005), Logendran *et al.* (2007), Anghinolfi e Paolucci (2007).

Parker *et al.* (1977) formulam o problema de programação em máquinas paralelas como um problema de roteamento de veículos e desenvolvem algoritmos baseados no clássico algoritmo construtivo para roteamento de veículos proposto por Clarke e Wright (1964) para minimizar o custo total de preparação sujeito a restrições de capacidade.

Guinet (1991) desenvolve heurísticas construtivas para analisar duas medidas de desempenho – tempo médio de conclusão das tarefas e atraso médio – em ambientes de máquinas não relacionadas. Em Guinet (1993) as medidas de desempenho consideradas são a minimização do máximo instante de conclusão (*makespan*) e a minimização do tempo de conclusão médio de um conjunto de tarefas a serem alocadas em máquinas idênticas. Em Guinet (1995) as máquinas são uniformes e as medidas de desempenho são duas – minimização do atraso médio e da soma ponderada dos atrasos. Nos três trabalhos Guinet considera tarefas disponíveis no instante zero e distintas datas de entrega para os casos de minimização de atraso.

Ovacik e Uzsoy (1995) desenvolvem uma família de heurísticas de horizontes rolantes para fazer a programação de tarefas com distintos instantes de liberação e distintas datas de entrega, em máquinas paralelas idênticas, a fim de minimizar o *lateness* máximo.

França *et al.* (1996) avaliam a minimização de *makespan* em máquinas paralelas idênticas, com todas as tarefas disponíveis no instante zero. A metodologia adotada apresenta três fases. Na primeira, uma solução de partida é obtida pela aplicação de uma heurística construtiva gulosa, tal que a cada iteração é alocada a tarefa que contribui com o menor aumento para o *makespan*. Na segunda é aplicada uma busca local em que as tarefas são movidas, uma por vez, de uma máquina para outra. O procedimento é controlado por uma Busca Tabu. Por fim, na terceira fase, procede-se a uma pós-otimização da máquina mais carregada.

Lee e Pinedo (1997) abordam o problema de minimização de soma ponderada de atrasos em máquinas idênticas. A metodologia de solução combina a regra de despacho ATCS (*Apparent Tardiness Cost with Setups*) para obtenção de uma solução inicial, seguida da aplicação de *Simulated Annealing*.

Park *et al.* (2000) estudam o mesmo problema investigado por Lee e Pinedo (1997), e sugerem um novo parâmetro, denominado faixa dos tempos de preparação, para caracterização das instâncias. Uma rede neural é empregada para estabelecer relações mais apropriadas entre os parâmetros de caracterização das instâncias e os fatores de *look-ahead* k_1 e k_2 da regra ATCS. Testes computacionais indicam um desempenho cerca de 6% superior em relação à forma original de cálculo dos fatores de *look-ahead*.

Gendreau *et al.* (2001) investigam a minimização de *makespan*, em máquinas paralelas idênticas, por meio de uma metodologia constituída por duas fases. Na primeira, uma solução inicial é obtida resolvendo-se o problema de atribuição associado ao problema de minimização de *makespan*. Nesta fase, os tempos de processamento são modificados pela inclusão dos tempos de preparação. Na segunda fase a solução é melhorada por meio de uma busca local em que são utilizados movimentos de blocos de tarefas de tamanhos variados.

Kurz e Askin (2001) avaliam o desempenho de algumas heurísticas construtivas e também de uma implementação de algoritmos genéticos para minimizar o *makespan* em máquinas paralelas idênticas. Cada tarefa está liberada para processamento em um instante específico e os tempos de preparação podem ser simétricos ou não, dependendo do cenário estudado, mas sempre respeitando a desigualdade triangular.

Weng *et al.* (2001) desenvolvem uma série de heurísticas construtivas para o problema de minimização do tempo total ponderado de conclusão de um conjunto de tarefas, disponíveis no instante zero, em máquinas não relacionadas.

Hiraishi *et al.* (2002) desenvolvem estudos teóricos sobre a maximização do número ponderado de tarefas concluídas nas respectivas datas de entrega. As máquinas são idênticas e as tarefas estão todas disponíveis no instante zero. Os tempos de preparação respeitam a desigualdade triangular. Os autores mostram que este problema pode ser resolvido em tempo polinomial. Contudo, se em lugar de datas de entrega definidas forem estabelecidas janelas de tempo para conclusão das tarefas, o problema passa a ser *NP-hard*, mesmo para o caso de uma única máquina com penalizações unitárias e sem tempos de preparação.

Kim e Shin (2003) investigam a minimização do máximo *lateness* em ambientes de máquinas idênticas e também de máquinas não relacionadas. A metodologia de solução envolve uma regra de despacho que emprega um índice de prioridade baseado no índice ATCS (Lee *et al.*, 1997) mas que leva em consideração o fato das tarefas terem distintos instantes de liberação. O índice proposto é denominado MATCS (*Modified Apparent Tardiness Cost with Setups*). A partir da solução inicial é aplicada uma Busca Tabu com movimentos de inserção e troca de pares de tarefas.

Fowler *et al.* (2003) desenvolvem um procedimento híbrido para problemas de programação em máquinas idênticas. As medidas de desempenho consideradas são três: *makespan*, tempo de conclusão total ponderado, e atraso total ponderado. O procedimento é formado por uma implementação de algoritmos genéticos para promover a atribuição das tarefas às máquinas e de uma regra de despacho, originalmente formulada para problemas de uma única máquina, para efetuar o seqüenciamento em cada máquina.

Bilge *et al.* (2004) consideram o problema de minimização do atraso total em máquinas uniformes. A metodologia utilizada consiste da obtenção de uma solução inicial por meio da regra EDD (*earliest due date*) seguida da aplicação de Busca Tabu, com movimentos de inserção e troca de pares de tarefas. Uma estratégia dinâmica e sistemática envolvendo uma seqüência de pequenas, médias e longas durações tabu é utilizada para promover um balanço entre diversificação e intensificação no curto prazo. A estratégia de diversificação de longo prazo consiste em aumentar significativamente a duração tabu, após ter sido realizado um número pré-especificado de movimentos sem melhoria da solução incumbente. A estratégia de intensificação de longo prazo, também ativada após a realização de um número pré-especificado de movimentos sem melhoria da solução incumbente, consiste em apagar os registros de memória e reiniciar a busca de curto prazo a partir da melhor solução de um pequeno conjunto de elite.

Anglani *et al.* (2005) investigam a minimização de custos de preparação em um ambiente de máquinas paralelas idênticas com incertezas nos tempos de processamento.

Rojanasoonthon e Bard (2005) desenvolvem uma implementação de GRASP (*Greedy Randomized Adaptive Search Procedure*) para resolver o problema de maximização da soma ponderada de tarefas alocadas em máquinas paralelas não relacionadas. A cada tarefa são designadas duas janelas de tempo disjuntas, devendo a alocação ocorrer, necessariamente, em uma das janelas.

Logendran *et al.* (2007) estudam a minimização da soma ponderada de atrasos em máquinas paralelas não relacionadas, em um contexto no qual tarefas e máquinas podem não estar disponíveis no instante inicial do horizonte de planejamento. O problema é resolvido com uma Busca Tabu em que são empregados movimentos de troca e de inserção. Freqüências de residência de alocação são utilizadas para induzir diversificação da busca.

Anghinolfi e Paolucci (2007) desenvolvem uma metaheurística híbrida, com elementos de Busca Tabu, *Simulated Annealing* e *Variable Neighborhood Search* para resolver o mesmo problema abordado por Bilge *et al.* (2004), tendo conseguido resultados melhores.

2.3 GRASP

GRASP (*Greedy Randomized Adaptive Search Procedure*) é um procedimento de múltiplos reinícios e como tal é caracterizado pela realização de iterações compostas pela construção de uma solução inicial orientada por uma função gulosa, seguida de uma busca local para a obtenção de um ótimo local. As iterações são repetidas até que seja satisfeito um critério de parada e o melhor ótimo local é selecionado. A lista de aplicações bem sucedidas de GRASP inclui uma grande variedade de problemas combinatórios (Feo e Resende, 1995, Festa e Resende, 2004). Neste trabalho são propostas versões de GRASP em que são empregadas abordagens adaptativas baseadas em memória. Outras técnicas desenvolvidas para melhorar o desempenho do GRASP básico são discutidas em Resende e Ribeiro (2003).

2.3.1 Função gulosa

Na fase construtiva, o próximo elemento a ser incorporado à solução parcial é escolhido probabilisticamente a partir de uma lista restrita de candidatos, que contém os melhores elementos segundo uma função de avaliação gulosa. A função gulosa usada para o problema **PST** é baseada no índice de prioridade ATCS (Lee *et al.*, 1997), descrito a seguir.

Seja t^{ϕ} o instante de liberação da máquina ϕ, isto é, o instante de conclusão da última tarefa alocada em ϕ. Seja t^* o menor dentre todos os instantes de liberação e seja ϕ^* a máquina liberada mais cedo, isto é, liberada no instante t^*. Se houver mais de uma máquina liberada no instante t^*, uma delas é escolhida aleatoriamente, com probabilidade uniforme. Seja k a última tarefa alocada na máquina ϕ^*. Seja j uma tarefa pertencente ao conjunto de tarefas não alocadas, $\overline{N}$. Cada tarefa j é caracterizada por:

- tempo de processamento: p_j
- custo por unidade de tempo de atraso: t_j
- data de entrega: d_j
- tempo de preparação para execução imediatamente após a tarefa k: s_{kj}

Além disso, considere que $\overline{p}$ e $\overline{s}$ representam o tempo de processamento médio e o tempo de preparação médio das tarefas não alocadas. No cálculo de $\overline{s}$ são levados em conta apenas os tempos de preparação s_{ki} associados à transição entre a última tarefa alocada na máquina ϕ*, tarefa k, e as tarefas não alocadas, bem como os tempos de preparação entre as próprias tarefas não alocadas, ou seja, os tempos de preparação s_{ij} e s_{ji}, em que i e $j \in \overline{N}$. Tempos de preparação entre as tarefas já alocadas não exercem influência no cálculo do índice de prioridade e por este motivo são desconsideradas no cálculo de $\overline{s}$.

Para cada tarefa não alocada j, o índice de prioridade ATCS é calculado segundo a Equação 2.1, em que k_1 e k_2 são fatores de escala dados pelas equações 2.2 e 2.3, em função de parâmetros característicos da instância, conforme sugerido por Lee *et al.* (1997).

$$I_j\left(t^*, k\right) = \frac{t_j}{p_j}\exp\left(-\frac{\max((d_j - p_j - t^*),\ 0)}{k_1\overline{p}}\right)\exp\left(-\frac{s_{kj}}{k_2\overline{s}}\right) \tag{2.1}$$

No índice ATCS, a primeira exponencial atribui prioridade máxima para as tarefas atrasadas ($d_j - p_j - t^* < 0$), independentemente de quão atrasadas elas estejam. Para as demais, a prioridade atribuída é tanto menor quanto mais longe a data de entrega estiver do instante t^* de liberação da máquina ϕ^*. Já a segunda exponencial prioriza tarefas que exigem menores tempos de preparação para execução imediatamente após a tarefa k.

$$\begin{aligned} &k_1 = 1.2\ln(\mu) - R \\ &\quad \text{se} \quad \tau < 0.5, \quad k_1 = k_1 - 0.5 \\ &\quad \text{se} \quad \eta < 0.5 \quad \text{e} \quad \mu > 5, \quad k_1 = k_1 - 0.5 \end{aligned} \tag{2.2}$$

$$\begin{aligned} &k_2 = \frac{\tau}{A_2\sqrt{\eta}} \\ &\quad A_2 = 1.8 \quad \text{se} \quad \tau < 0.8 \\ &\quad A_2 = 2.0 \quad \text{se} \quad \tau > 0.8 \end{aligned} \tag{2.3}$$

Os parâmetros característicos utilizados nas equações 2.2 e 2.3 são:

- Faixa das datas de entrega, $R = (d_{\max} - d_{\min}) / C_{\max}$;
- Fator de aperto das datas de entrega, $\tau = 1 - (\bar{d} / C_{\max})$;
- Fator de severidade dos tempos de preparação, $\eta = \bar{s} / \bar{p}$;
- Relação entre tarefas e máquinas, $\mu = n / m$

Nos parâmetros anteriores, $\bar{d}, d_{\min}, d_{\max}$ são, respectivamente, os valores médio, mínimo e máximo das datas de entrega. C_{max}, uma estimativa do *makespan*, é calculado por $C_{\max} = (\beta \bar{s} + \bar{p})\mu$, em que $\beta = 0.4 + \frac{10}{\mu^2} - \frac{\eta}{7}$, para $\mu \geq 5$.

Tanto no cálculo da estimativa do *makespan*, quanto no cálculo do fator de severidade dos tempos de preparação, empregam-se valores de $\bar{s}$ e $\bar{p}$ que têm por base o conjunto de todas as tarefas. Portanto, são valores calculados uma única vez e característicos da instância. Já no cálculo do índice de prioridade, Equação 2.1, $\bar{s}$ e $\bar{p}$ são atualizados a cada nova tarefa alocada.

Denote por $I_{\max}$ o maior índice de prioridade no instante t^*, dentre todas as tarefas não alocadas. A função gulosa é definida na Equação 2.4 e a lista restrita de candidatas é composta pelas tarefas cujos valores da função satisfaçam a Equação 2.5, em que $g_{\min}$ e $g_{\max}$ representam, respectivamente, o menor e o maior valor de $g_j(t^*, k)$.

$$g_j(t^*, k) = \frac{I_j(t^*, k)}{I_{\max}} \tag{2.4}$$

$$g_j(t^*, k) \in \left[g_{\min} + (1 - \alpha)(g_{\max} - g_{\min}) \quad , \quad g_{\max} \right] \tag{2.5}$$

Na Equação 2.5, o parâmetro $\alpha \in [0,1]$ serve para controlar o tamanho da lista restrita de tarefas candidatas. Assim, $\alpha = 1$ corresponde a uma lista irrestrita, que contém todas as tarefas não alocadas. Por outro lado, quando $\alpha = 0$ a lista tem, em geral, tamanho unitário. Neste último caso, a escolha da próxima tarefa a alocar é puramente gulosa. O valor de α é

escolhido aleatoriamente, com probabilidade uniforme, a cada iteração (fase construtiva + busca local). Por fim, a probabilidade de escolha de uma tarefa da lista restrita de candidatas (LRC), a cada passo do processo de construção da solução inicial é definida pela Equação 2.6. A probabilidade de escolha é diretamente proporcional ao valor da função gulosa. Outras distribuições de probabilidade foram sugeridas por Bresina (1996) e utilizadas por Binato *et al.* (2002).

$$p_j(t^*,k) = \frac{g_j(t^*,k)}{\sum_{r \in LRC} g_r(t^*,k)} \tag{2.6}$$

É importante ressaltar que a função gulosa precisa apresentar valores positivos, para que seja possível estabelecer distinções entre as diversas tarefas não alocadas. A função objetivo, baseada em atrasos, é, desse modo, imprópria para utilização como função gulosa, uma vez que existe a possibilidade da alocação de uma ou mais tarefas em uma solução parcial ocorrer sem um incremento no custo da solução.

2.3.2 Busca local

São investigados movimentos comumente utilizados em problemas de programação de tarefas, bem como movimentos utilizados em problemas de roteamento de veículos. Deve ser observado que o problema **PST** pode ser visto como o problema de roteamento assimétrico, sem restrições de capacidade. A distância entre dois clientes corresponde à soma do tempo de processamento de uma tarefa e o tempo de preparação para a tarefa subseqüente. Os clientes devem ser atendidos ao longo de uma janela de tempo com extremos que correspondem ao instante de liberação e a data de entrega da tarefa. Os veículos não devem retornar ao depósito e a função objetivo consiste em minimizar o atraso total ponderado em relação às datas de entrega impostas pelos clientes. Parker *et al.* (1977) consideram este ponto de vista para propor uma extensão da heurística construtiva clássica proposta por Clarke e Wright (1964) para o problema de roteamento de veículos. França *et al.* (1996) usam adaptações de heurísticas de inserção desenvolvidas por Gendreau *et al.* (1992) para o problema do caixeiro viajante simétrico.

Os movimentos clássicos usados em problemas de programação de tarefas consistem em trocas entre pares de tarefas e inserções de uma única tarefa. Podem estar restritos a uma única máquina (trocas e inserções internas), assim como podem afetar duas máquinas simultaneamente (trocas e inserções externas). Esses movimentos induzem a formação das vizinhanças de troca e de inserção.

Em termos de movimentos comumente utilizados em roteamento de veículos, o presente trabalho está concentrado nos movimentos *cross* (Taillard *et al.* 1997) e *Or-Opt* (Or, 1976) – outros movimentos para roteamento de veículos podem ser vistos em Savelsbergh (1992).

O movimento *Or-Opt* , mostrado na Figura 2.2, é uma forma restrita do *3-Opt* e envolve a inserção de blocos compostos por até 3 tarefas consecutivas em outras posições da própria máquina ou de outras máquinas. Tal movimento é usado em meta-heurísticas (Bräysy, 2003; Homberger e Gehring, 2005; Bräysy e Gendreau, 2005) para roteamento de veículos com janelas de tempo porque explorar sua vizinhança completa requer um esforço computacional $O(n^2)$ contra um esforço $O(n^3)$ associado à exploração da vizinhança completa *3-Opt* e com desempenho similar. No exemplo mostrado na Figura 2.2, o bloco composto pelas tarefas 5 e 10 é inserido na última posição, após a tarefa 4.

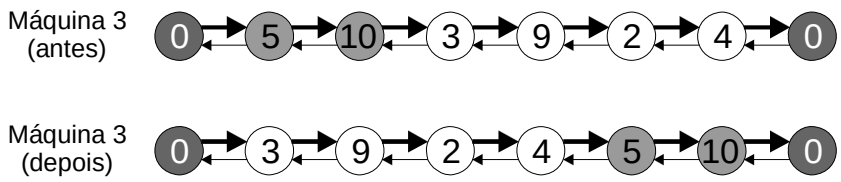

Figura 2.2 – Movimento *Or-Opt*

Os movimentos *cross* promovem trocas de blocos de tarefas entre duas máquinas. Os blocos são compostos por tarefas adjacentes e podem ter tamanhos variados. Quando ambos os blocos são formados por apenas uma tarefa, os movimentos são idênticos àqueles de troca de pares de tarefas entre máquinas. Quando um dos blocos tem tamanho nulo e o outro tamanho unitário, tem-se os mesmos movimentos da vizinhança de inserção entre máquinas. Se um dos blocos tiver tamanho nulo e o outro for composto por até três tarefas, os movimentos

obtidos correspondem aos movimentos *Or-Opt* entre máquinas. Por fim, se cada bloco contiver a última tarefa de cada máquina, os movimentos *cross* serão iguais aos movimentos *2Opt** (Potvin e Rousseau, 1995). No presente trabalho os blocos são compostos por até três tarefas consecutivas. Pelo fato da vizinhança *cross* conter a vizinhança *Or-Opt* entre máquinas, esta última não é explorada. Enquanto os movimentos *cross* desempenham o papel de alterar a atribuição de tarefas às máquinas e também de alterar o total de tarefas atribuídas a cada máquina, os movimentos *Or-Opt* internos são explorados buscando-se uma otimização local em cada máquina. Na Figura 2.3 é mostrado um exemplo de movimento *cross*, no qual o bloco de tarefas adjacentes 7 e 8, originalmente na máquina 1, é trocado com o bloco de tarefas 4 e 1 (máquina 2).

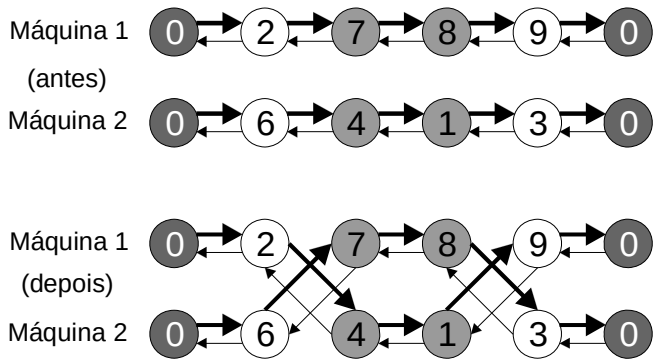

Figura 2.3 – Movimento *cross*

Na Figura 2.4 mostra-se o algoritmo geral do GRASP com buscas locais nas vizinhanças de troca e de inserção, exploradas de modo alternado. Uma vez definida uma solução inicial, efetua-se a busca local na vizinhança de troca (VT) que divide-se em trocas internas (VTi) e trocas externas (VTe). Movimentos em VTi são executados até que um mínimo local em cada máquina seja obtido. Passa-se, então, a explorar a vizinhança VTe, que tem como ponto de partida a solução composta pelos mínimos locais em cada máquina. Se um novo mínimo local for alcançado com movimentos de troca entre máquinas, retoma-se a exploração dos movimentos VTi. Caso contrário, passa-se à exploração da vizinhança de inserção (VI), dividida em inserções internas (VIi) e inserções externas (VIe), de modo semelhante àquele descrito para a vizinhança de troca. Prossegue-se com a alternância de buscas locais até que seja alcançado um mínimo local em relação a ambas as vizinhanças (VT e VI) ou até que o critério de parada seja satisfeito.

1. Leia os parâmetros da instância.
Z(Incumbente) = infinito
2. Repita os passos 3 a 5 até que o critério de parada seja satisfeito.
3. Fase construtiva.
Sol_VT = Restart ( )
Z(Sol_VI) = infinito
4. Busca local de troca
Sol_VT$_i$ = Sol_VT
Z (Sol_VTe) = infinito
a) Para cada uma das máquinas, aplique busca local sobre Sol_VT$_i$
Se Z (Sol_VT$_i$) < Z (Incumbente) Incumbente = Sol_VT$_i$
Se Z (Sol_VT$_i$) < Z (Sol_VT$_e$)
Sol_VT$_e$ = Sol_VT$_i$ e vá para o passo 4b.
Caso contrário, vá para o passo 4c
b) Aplique busca local sobre Sol_VT$_e$ (trocas externas)
Se Z (Sol_VT$_e$) < Z (Incumbente) Incumbente = Sol_VT$_e$
Se Z (Sol_VT$_e$) < Z (Sol_VT$_i$)
Sol_VT$_i$ = Sol_VT$_e$ e vá para o passo 4a.
Caso contrário, vá para o passo 4c
c) Sol_VT = Sol_VTe
Se Z (Sol_VT) < Z (Sol_VI)
Sol_VI = Sol_VT e vá para o passo 5
Caso contrário, vá para o passo 3
5. Busca local de inserção
Sol_VI$_i$ = Sol_VI
Z (Sol_VIe) = infinito
a) Para cada uma das máquinas, aplique busca local sobre Sol_VI$_i$
Se Z (Sol_VI$_i$) < Z (Incumbente) Incumbente = Sol_VI$_i$
Se Z (Sol_VI$_i$) < Z (Sol_VI$_e$)
Sol_VI$_e$ = Sol_VI$_i$ e vá para o passo 5b.
Caso contrário, vá para o passo 5c
b) Aplique busca local sobre Sol_VI$_e$ (inserções externas)
Se Z (Sol_VI$_e$) < Z (Incumbente) Incumbente = Sol_VI$_e$
Se Z (Sol_VI$_e$) < Z (Sol_VI$_i$)
Sol_VT$_i$ = Sol_VT$_e$ e vá para o passo 5a.
Caso contrário, vá para o passo 5c
c) Sol_VI = Sol_VIe
Se Z (Sol_VI) < Z (Sol_VT)
Sol_VT = Sol_VI e vá para o passo 4
Caso contrário, vá para o passo 3

Figura 2.4 – Algoritmo geral do procedimento GRASP: Vizinhanças de troca e de inserção

Na Figura 2.5 mostra-se o algoritmo geral do GRASP com buscas locais nas vizinhanças *cross* e *Or-Opt*, exploradas de modo alternado. Começa-se com a busca local na vizinhança *cross* (VC). O mínimo local obtido ao final é tomado como ponto de partida para a busca local na vizinhança *Or-Opt* (VOr). Prossegue-se com a alternância de buscas locais até que seja alcançado um mínimo local em relação às duas vizinhanças, ou até que o critério de parada seja satisfeito.

1. Leia os parâmetros da instância.
Z (Incumbente) = infinito
2. Repita os passos 3 a 5 até que o critério de parada seja satisfeito.
3. Fase construtiva.
Sol_VC = Restart ( )
Z (Sol_VOr) = infinito
4. Aplique busca local sobre Sol_VC.
Se Z (Sol_VC) < Z (Incumbente) Incumbente = Sol_VC.
Se Z (Sol_VC) < Z (Sol_VOr)
Sol_VOr = Sol_VC e vá para o passo 5.
Caso contrário, vá para o passo 3
5. Aplique busca local sobre Sol_VOr.
Se Z (Sol_VOr) < Z (Incumbente) Incumbente = Sol_VOr.
Se Z (Sol_VOr) < Z (Sol_VC)
Sol_VC = Sol_VOr e vá para o passo 4.
Caso contrário, vá para o passo 3

Figura 2.5 – Algoritmo geral do procedimento GRASP: Vizinhanças *cross* e *Or-Opt*

Redução de vizinhança

A avaliação das vizinhanças completas requer um grande e muitas vezes injustificável esforço computacional. Por esse motivo é aplicada uma estratégia de redução de vizinhancas que tem por objetivo evitar a avaliação de soluções que apresentem determinadas características julgadas ruins. Como conseqüência, mais vizinhanças podem ser exploradas. No caso presente da programação de tarefas, espera-se que soluções de boa qualidade envolvam tempos de preparação eficientes, muito embora a função objetivo do problema não seja a minimização dos tempos de preparação. Quando qualquer movimento é realizado, ocorrem mudanças no conjunto de tempos de preparação associados às relações de precedência

desfeitas, $\sum s_{ij(antigos)}$, e criadas pelo movimento, $\sum s_{ij(novos)}$. Assim, são avaliadas apenas as soluções que satisfazem a expressão $\sum s_{ij(antigos)} \geq \xi \sum s_{ij(novos)}$, em que $\xi \geq 0$ é um parâmetro a ser calibrado. $\xi = 0$ corresponde à exploração da vizinhança completa. Valores de $\xi > 0$ impõem uma redução na vizinhança, tão mais intensa quanto maior o valor de ξ .

2.3.3 Estratégias para melhoria do desempenho do GRASP básico

Duas abordagens baseadas no uso de memória de longo prazo foram propostas na literatura com o objetivo de melhorar o desempenho do GRASP básico.

Fleurent e Glover (1999) apresentam uma integração entre GRASP e memória adaptativa, por meio de uma combinação entre a função gulosa usada na fase construtiva e uma função de intensificação, cuja influência é tanto maior quanto mais freqüente for a ocorrência de um determinado atributo nas soluções do conjunto de elite e quanto menores forem os custos das soluções de elite nas quais tais atributos estiverem presentes. Desse modo, as probabilidades de escolha dos elementos da lista restrita de candidatos são modificadas e as soluções iniciais passam a adquirir características presentes nas soluções de elite. Esta abordagem foi utilizada no problema de *job shop scheduling* (Binato *et al.*, 2002), no problema de *clustering* com restrições de capacidade (Ahmadi e Osman, 2005) e no seqüenciamento de DNA por hibridização (Fernandes e Ribeiro, 2005).

Laguna e Martí (1999) usam memória de longo prazo de uma maneira diferente. Ao final de cada iteração realiza-se uma religação de caminho – *path relinking* – (Glover, 1996) para conectar o mínimo local do GRASP a uma solução do conjunto de elite escolhida aleatoriamente.

Resende e Ribeiro (2005) enfatizam a importância da aplicação de religação de caminho para a melhoria do desempenho do GRASP básico e indicam duas maneiras para fazê-la. A primeira é aquela apresentada por Laguna e Martí (1999). A segunda é como uma estratégia de pós-otimização, na qual as soluções de elite são conectadas entre si.

O desempenho do GRASP básico também pode ser melhorado pela aplicação do princípio de otimalidade próxima (Glover e Laguna, 1997), em que a solução parcial é melhorada por meio de busca local antes que um novo elemento seja a ela adicionado.

Fase construtiva com memória

A lista restrita de tarefas candidatas, no GRASP com memória, é definida sob a influência de uma população de soluções de elite, as quais correspondem a soluções de alta qualidade e suficientemente distintas. Constrói-se uma função de avaliação, $E(t^*,j)$, para cada tarefa j não alocada, a partir de uma combinação linear entre a função gulosa (definida no item 2.3.1) e uma função de intensidade, que representa uma medida das características das soluções de elite. Pesos são dados a cada uma de tais funções, de modo a controlar o grau de diversificação / intensificação ao longo das iterações.

A função de intensidade é uma medida da consistência dos atributos da solução (Glover e Laguna, 1997; Fleurent e Glover, 1999). Em problemas de programação de tarefas está relacionada a aspectos de atribuição das tarefas às máquinas e de relações de precedência imediata entre tarefas. Para o problema **PST** aqui tratado apenas as relações de precedência imediata entre tarefas são consideradas, uma vez que as máquinas são idênticas. A função de intensidade aumenta à medida que estes atributos aparecem com maior freqüência nas soluções de elite.

Seja ϕ a máquina na qual deve ser inserida a próxima tarefa. Seja k a última tarefa alocada em ϕ. Para cada tarefa não alocada j, calcula-se a função de intensidade $H_k(j)$ que depende das relações de precedência entre as tarefas k e j nas soluções de elite.

Para problemas em que as máquinas são idênticas a função de intensidade é dada pela Equação 2.7, em que é levado em consideração apenas o aspecto de precedência entre tarefas, visto que neste tipo de problema é irrelevante a associação tarefa-máquina.

$$H_k(j) = \sum_{S \in Elite} \delta(S,k,j) \frac{Z(S^*)}{Z(S)} \quad (2.7)$$

Na Equação 2.7, $\delta(S,k,j) = 1$ se a tarefa k precede a tarefa j na solução de elite S. Caso contrário, $\delta(S,k,j) = 0$. $Z(S)$ representa o custo da solução de elite S, enquanto que $Z(S^*)$ representa o custo da melhor solução de elite.

Antes de ser combinada com a função gulosa, a função de intensidade é normalizada em termos de $H_{\max}$, o máximo valor de $H_k(j)$, dando origem à função $h(j)$, como indicado na Equação 2.8.

$$h(j) = \frac{H_k(j)}{H_{\max}} \quad (2.8)$$

A função de avaliação, $E(t^*, j)$, para qualquer tarefa j não alocada, é então definida como uma combinação convexa das funções gulosa (Equação 2.4) e de intensidade (Equação 2.8), ambas normalizadas, como indicado na Equação 2.9.

$$E(t^*, j) = (1-\lambda).g(t^*, j) + \lambda.h(j) \quad (2.9)$$

O fator de ponderação, λ, presente na Equação 2.9, assume valores no intervalo $[0,1]$ e controla o grau de intensificação ou de diversificação. Quando $\lambda = 0$ tem-se uma fase construtiva sem memória, que pode ser associada a uma diversificação. Por outro lado, quando $\lambda = 1$, a construção da solução inicial está totalmente orientada pela história da busca, correspondendo, portanto, a uma intensificação. A função de intensidade normalizada permite enfatizar a influência da memória durante a busca, como confirmado por estudos anteriores (Christofoletti, 2002).

A lista restrita de tarefas candidatas é formada, neste contexto, pelas tarefas que satisfazem a condição imposta pela Equação (2.10), em que $\alpha \in [0,1]$ é definido, a cada iteração, com probabilidade uniforme. Nesta equação, $E_{\min}$ e $E_{\max}$ representam, respectivamente o valor mínimo e máximo assumidos pela função de avaliação $E(t^*, j)$.

$$E(t^*, j) \in \left[E_{\min} + (1-\alpha)(E_{\max} - E_{\min}) \quad , \quad E_{\max}\right] \tag{2.10}$$

A probabilidade de escolha de uma tarefa da lista restrita de candidatas (LRC), a cada passo do processo de construção da solução inicial é agora definida na Equação 2.11.

$$p_j(t^*, k) = \frac{E(t^*, j)}{\sum_{i \in LRC} E_i(t^*, j)} \tag{2.11}$$

O controle do fator de ponderação λ é crucial para alcançar o correto balanço entre os estágios de diversificação e de intensificação. Por este motivo sua determinação é feita de maneira adaptativa, em função do grau de diversidade entre as soluções de mínimo local obtidas ao longo da busca.

Experimentos realizados por Armentano e França Filho (2007) mostram que a função de entropia normalizada, e, definida a partir das freqüências de precedências imediatas entre tarefas, ψ_{ij}, constitui uma função adequada para quantificar o grau de diversidade entre as soluções de um determinado conjunto.

Entropia é uma propriedade de qualquer distribuição de probabilidade, sendo uma medida básica em teoria da informação (Shannon, 1948). Jaines (2003) sugere que um nome mais apropriado seria *entropia da informação*, para distingüi-la da *entropia experimental*, que é uma propriedade de um estado termodinâmico. Entropia tem sido usada em algoritmos genéticos (Burke *et al.*, 2004) como uma medida da diversidade da população.

A função de entropia utilizada no presente trabalho é apresentada na Equação 2.12, para um conjunto de x_1 soluções obtidas ao final de x_1 iterações. A entropia pode assumir valores no intervalo $[0,1]$. Quando as x_1 soluções são todas idênticas, todos os elementos ψ_{ij}, para todo i diferente de j, ou são nulos ou assumem o valor $\psi_{ij} = x_1$. Desse modo, cada parcela do numerador fica multiplicada por $\log(1) = 0$. Portanto, tem-se $e = 0$. Por outro lado, se o número x_1 de soluções for um múltiplo do número n de tarefas e se ψ_{ij}, para todo i diferente de j, for igual a x_1/n, tem-se uma distribuição uniforme nas ocorrências de todas as possíveis relações de precedências entre tarefas. Logo, $e = 1$. Deve ser observado que o somatório na Equação 2.12 considera apenas os elementos para os quais $\psi_{ij} > 0$.

$$e = \frac{-\sum_{i=0}^{n}\sum_{j=1, j\neq i}^{n}\left(\psi_{ij}/x_1\right)\log\left(\psi_{ij}/x_1\right)}{n\log(n)} \tag{2.12}$$

Nas iterações iniciais, que formam a fase de aprendizado, o fator de ponderação λ é fixado em zero e o GRASP é executado a fim de construir um conjunto representativo de soluções de elite e a fim de definir uma medida de entropia de referência. A cada x_1 iterações a entropia é calculada e a matriz de freqüência de residência, cujos elementos são as freqüências de precedência imediata entre tarefas, ψ_{ij} , é reinicializada. A entropia média associada aos conjuntos de x_1 iterações da fase de aprendizado é denotada por $\bar{e}_a$. Ao final da fase de aprendizado o fator de ponderação λ assume um valor inicial λ_i e a memória começa a ser utilizada. Como na fase de aprendizado, a entropia e_{x1} para cada conjunto de x_1 iterações é calculada. Se $e_{x1} \geq \bar{e}_a$, o uso de memória não compromete a diversidade das soluções obtidas, em comparação à diversidade observada na fase de aprendizado. Então o fator λ é incrementado em 0.1 e um novo conjunto de x_1 iterações é realizado. Por outro lado, se $e_{x1} < \bar{e}_a$, há uma queda de diversidade em comparação àquela da fase de aprendizado. O fator λ é diminuído em 0.1 e um novo conjunto de x_1 iterações é realizado. Quando o fator λ assume um de seus valores extremos ($\lambda_{\min}$ ou $\lambda_{\max}$) por um determinado número de iterações, o valor do outro extremo é ativado, forçando a mudança de um estágio de diversificação para um estágio de intensificação ou vice-versa. Na Figura 2.6 é esboçado o procedimento de atualização do fator de ponderação λ.

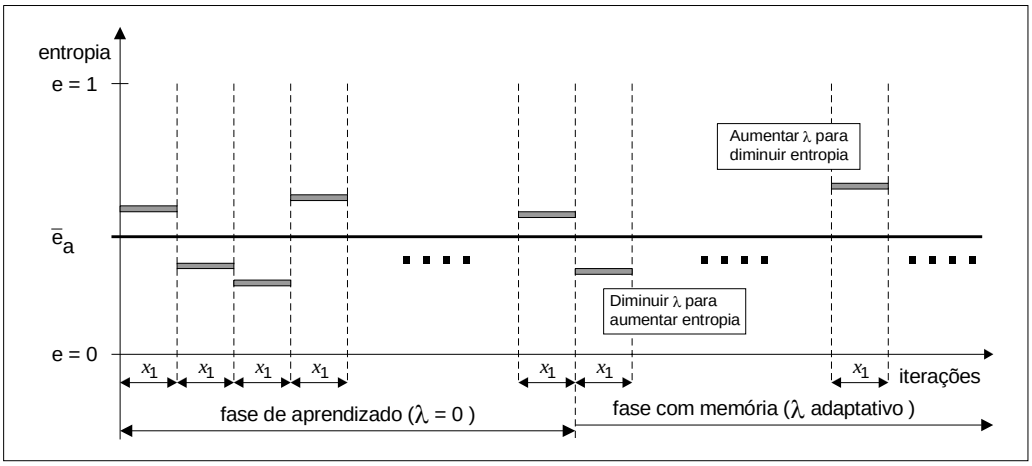

Figura 2.6 – Ajuste adaptativo de λ, ao longo das iterações, em função da entropia

O conjunto de elite, referido anteriormente, é composto por soluções de alta qualidade e suficientemente distintas entre si. A qualidade de uma solução é avaliada pelo seu custo, enquanto que o grau de diversidade entre duas soluções é dado pela distância entre elas (definida mais adiante).

Os parâmetros que caracterizam o conjunto de elite (Γ) são o número de soluções a serem armazenadas e a distância mínima entre elas ($D_{\min}$). No procedimento clássico (Fleurent e Glover, 1999), o conjunto é inicializado com todas as suas soluções (S_e) vazias e com custos $Z(S_e)$ infinitos. As variáveis $Z_{\max}$ e $Z_{\min}$, representando os custos da pior e da melhor solução presentes no conjunto de elite são inicializadas em infinito. A solução S_c obtida ao final da primeira iteração GRASP é inserida no conjunto Γ , substituindo uma solução vazia e $Z_{\min}$ é atualizado. As soluções S_c obtidas a partir da segunda iteração GRASP passam pela verificação de critérios de custo e de distância para entrarem no conjunto de elite. Se $Z(S_c) > Z_{\max}$ a solução candidata é descartada. Se $Z(S_c) < Z_{\max}$, então S_c é uma candidata a entrar para o conjunto Γ , desde que ela seja suficientemente distante das soluções S_e não vazias presentes no referido conjunto, ou seja, $D(S_c,S_e) > D_{\min} \ \forall \ S_e \in \Gamma$. Satisfeito o critério de distância, S_c substitui a solução S_e de pior custo. $Z_{\max}$ e $Z_{\min}$ são atualizados. A solução é descartada caso seja considerada muito próxima de alguma outra do conjunto de elite e caso o critério de aspiração, caracterizado por $Z(S_c) < Z_{\min}$, não seja satisfeito. Se o critério de aspiração for satisfeito, S_c entra para o conjunto de elite, qualquer que seja sua distância em relação às demais, substituindo a pior solução do conjunto.

Uma forma alternativa para atualização do conjunto de elite, utilizada em Armentano e França Filho (2007) é apresentada a seguir. Nela utiliza-se um critério flexível para aceitação da solução candidata em termos de sua distância em relação às soluções presentes no conjunto de elite. A solução é aceita se

$$\min_{\forall S_e \in \Gamma} D(S_c, S_e) > \max\left(\theta, \left[\frac{Z(S_C) - Z_{\min}}{Z_{\max} - Z_{\min}}\right]\right) D_{\min} .$$

O limiar de distância, $D_{\min}$, empregado no modelo clássico como um valor fixo, passa a ser multiplicado por um fator de escala que depende do custo da solução candidata e dos valores atuais de $Z_{\max}$ e $Z_{\min}$. Para a definição de $Z_{\max}$ não são consideradas as soluções vazias, que, por definição, têm custo infinito. Em todas as situações nas quais a solução candidata preenche os requisitos para ingressar no conjunto de elite, retira-se a solução de maior custo. Este modelo confere maior mobilidade ao conjunto de elite, permitindo a entrada de soluções de boa qualidade mas que estejam um pouco mais perto do que o valor $D_{\min}$ arbitrado e/ou a entrada de soluções que, embora um pouco mais caras que a pior solução de elite, encontram-se em regiões mais distantes do espaço de soluções. Em testes computacionais (Armentano e França Filho, 2007) aplicados às instâncias consideradas em Bilge *et al.* (2004), os resultados obtidos ao se atualizar o conjunto de elite segundo este procedimento mais flexível foram um pouco melhores que aqueles obtidos com o procedimento clássico de atualização.

Deve ser observado que o parâmetro θ, contido no intervalo $[0,1]$, impõe um limite à flexibilidade de aceitação da solução candidata. O critério ora proposto fica mais rígido e mais parecido com o critério clássico, quanto maior o valor de θ.

Dois atributos são considerados na definição de uma métrica para cálculo da distância entre duas soluções – a designação das tarefas e o seqüenciamento destas nas máquinas. Cada um destes aspectos é considerado em separado, dando origem às distâncias de designação e de seqüenciamento, que, uma vez normalizadas, são somadas de forma ponderada (Mazzini, 1998).

O conjunto de tarefas designadas a uma dada máquina pode ser interpretado como um grafo não direcionado em que os nós correspondem às tarefas e cada aresta ligando dois nós i e j implica no processamento das tarefas i e j por uma mesma máquina na solução em questão. A distância de designação (dd) entre duas soluções S_1 e S_2 é definida em termos dos totais de arestas em cada solução, respectivamente $n1$ e $n2$, e do número de arestas comuns às duas soluções, nc, da forma mostrada na Equação 2.13.

$$dd = \frac{1}{2}\left[\frac{(n1 - nc)}{n1} + \frac{(n2 - nc)}{n2}\right] \tag{2.13}$$

A relação de precedência imediata entre tarefas pode ser vista como a versão direcionada do grafo anterior, em que o arco (i,j) indica que a tarefa i precede imediatamente a tarefa j. A distância de seqüenciamento ou de arcos, (ds), é definida como sendo o número de arcos distintos dividido pelo total de arcos existentes em uma solução, que é constante e igual ao número de tarefas.

A distância total entre duas soluções é dada pela Equação 2.14. Nesta equação, observa-se que a distância de designação (dd) tem um peso que é tanto menor quanto menor for o número de máquinas da instância, até alcançar o valor zero em instâncias de máquina única. Por sua vez, o peso da distância de seqüenciamento (ds) é maior em instâncias em que há menos máquinas, como era de se esperar. Deve ser observado, ainda, que as distâncias de designação e de seqüenciamento variam no intervalo $[0,1]$. Logo, a distância total, D, varia no intervalo $[0,n]$.

$$D = \frac{n}{m}ds + \left(n - \frac{n}{m}\right)dd \tag{2.14}$$

Princípio de otimalidade próxima

O princípio de otimalidade próxima (POP) é baseado na idéia de que é bem provável que boas soluções em um nível estejam próximas a boas soluções em um nível adjacente (Glover e Laguna, 1997). Fleurent e Glover (1999) fornecem uma interpretação para este

princípio em um processo construtivo, em que decisões ruins em um estágio do processo tendem a levar a decisões ruins em estágios posteriores. Por esta razão, melhorias deveriam ser aplicadas em um determinado estágio, antes de prosseguir para o próximo. No contexto dos problemas tratados no presente trabalho, o princípio de otimalidade próxima pode ser traduzido na afirmação “boas soluções parciais com y tarefas são próximas a boas soluções parciais com $y+1$ tarefas”. Assim, o POP é aqui definido como buscas locais em soluções parciais.

Por questões de eficiência computacional, uma implementação prática do POP ao GRASP consiste em realizar buscas locais apenas em alguns poucos estágios da fase construtiva e não em todos eles. Tais pontos são definidos em termos de porcentagens de tarefas alocadas.

Religação de caminhos (*Path relinking*)

O procedimento de religação de caminhos (*Path relinking*) foi originalmente proposto por Glover (1996) como uma abordagem para integrar estratégias de intensificação e de diversificação em um contexto de Busca Tabu (Glover e Laguna, 1997) e Busca por Espalhamento (*Scatter Search*) (Glover, 1998; Martí *et al.*, 2006; Yamashita *et al.*, 2006). Esta abordagem gera novas soluções ao explorar trajetórias que conectam soluções de alta qualidade (intensificação) ou soluções de diferentes regiões do espaço de soluções ou que exibam aspectos contrastantes (diversificação).

Em sua forma mais simples a religação de caminhos envolve duas soluções de referência, denominadas solução inicial, S_i, e solução guia, S_g. Partindo da solução inicial, um caminho é gerado por meio de movimentos em uma dada vizinhança, em direção à solução guia. Em outras palavras, a religação de caminhos é um procedimento de busca em uma vizinhança modificada, constituída, em cada passo, por movimentos que introduzem atributos presentes na solução guia. Dentre os diversos movimentos possíveis executa-se o melhor, independentemente dele ser de melhoria ou de piora da função objetivo.

Neste trabalho a religação de caminhos é aplicada para conectar as soluções do conjunto de elite, obtidas ao longo das iterações GRASP, como uma estratégia de pós-otimização. Durante o procedimento pode surgir alguma solução que leve à atualização do

conjunto de elite. Neste caso, o conjunto é atualizado da forma descrita anteriormente e a religação de caminhos é aplicada para conectar cada uma das soluções de elite remanescentes à nova solução de elite.

Em um contexto de programação de tarefas em máquinas paralelas, uma implementação de religação de caminhos pode consistir, numa primeira etapa, em promover alterações ou correções nas atribuições das tarefas às máquinas, com as atribuições da solução guia tomadas como referência. Nesta etapa executam-se movimentos entre máquinas distintas. Neste trabalho são avaliadas duas variantes, denominadas RC1 e RC2, que se distinguem pelo tipo de movimento entre máquinas. Na variante RC1 a designação é alterada por movimentos de inserção. Na variante RC2 são executados movimentos *cross*. Na segunda etapa cada máquina é considerada individualmente, com movimentos que alteram os posicionamentos das tarefas. São empregados movimentos de inserção e/ou troca, prevalecendo o melhor dentre eles. Buscas locais são realizadas ao longo da trajetória em direção à solução guia, nas duas etapas, sempre que o custo da solução intermediária for menor que o custo da solução inicial e/ou menor que o custo da solução guia. Na variante RC1 as buscas locais empregam movimentos de troca e inserção, enquanto que na variante RC2 os movimentos empregados são *cross* e *Or-Opt*.

Para ilustrar o procedimento, considere uma instância com 10 tarefas e 3 máquinas. Assuma que a solução inicial seja dada por $S_i = \{(1,2,3),(4,5,6),(7,8,10,9)\}$ e que a solução guia seja $S_g = \{(7,1,4,5),(3,2,9),(10,8,6)\}$, em que cada parênteses contém a seqüência de tarefas processadas numa dada máquina – por exemplo, a máquina 1 da solução S_i processa as tarefas 1,2 e 3, nesta ordem, e assim por diante.

Na variante RC1, um dos possíveis movimentos de correção das atribuições de tarefas implica na retirada da tarefa 3 da máquina 1 e sua inserção na máquina 2. Como indicado na Figura 2.7, quatro novas soluções são definidas e avaliadas, uma para cada nova posição da tarefa 3. Seja $S_p = \{(1,2),(4,3,5,6),(7,8,10,9)\}$ a solução de menor custo. O número de tarefas "corretamente" atribuídas é aumentado em uma unidade. O procedimento é repetido a partir da solução S_p, até que a atribuição de cada tarefa em S_p seja igual àquela em S_g. Deve ser observado que eventualmente a distância (Equação 2.14) em relação à solução guia pode aumentar, como ocorre neste exemplo. Contudo, o procedimento proposto, quando comparado a outro que seja focado na diminuição da distância a cada passo, é de custo computacional mais baixo, (porque dispensa a verificação de distância) e gera uma trajetória mais rica, com mais possibilidades de movimento.

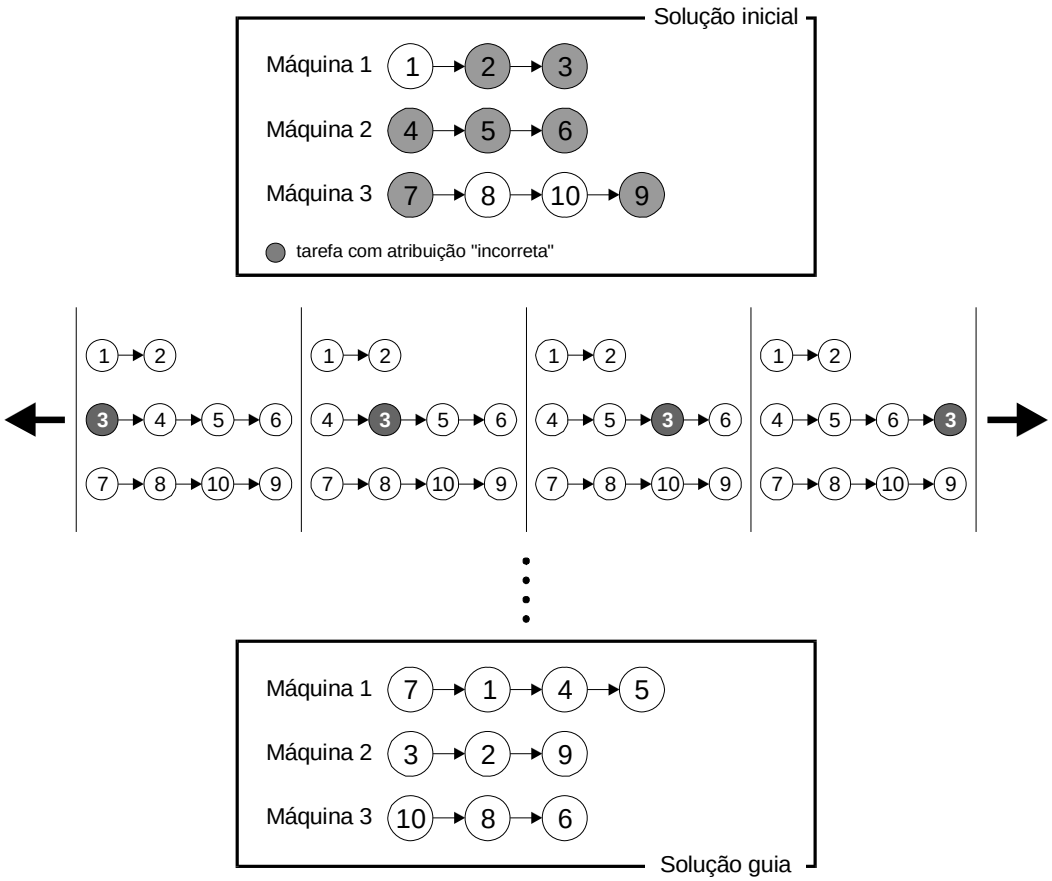

Figura 2.7 – Correção de atribuições na religação de caminho : Variante RC1

Na variante RC2, que emprega movimentos *cross*, um exemplo de movimento possível é aquele caracterizado pela troca da tarefa 1 (da máquina 1) com o bloco de tarefas adjacentes 4,5 (máquina 2). Associado a tal movimento são produzidas e avaliadas 6 soluções, como conseqüência das combinações de 3 soluções para os possíveis posicionamentos do bloco 4,5 na máquina 1 e 2 soluções para os possíveis posicionamentos da tarefa 1 na máquina 2. Tais soluções são mostradas na Figura 2.8. Em todas as soluções aumenta-se o número de tarefas corretamente atribuídas, apesar da tarefa 1 ter sido retirada da máquina “correta”.

Suponha que ao final da primeira etapa, de correção das atribuições, a solução obtida seja S_p = {(4,1,7,5),(3,9,2),(6,10,8)}. Na segunda etapa cada máquina é considerada em separado. Movimentos de inserção e de troca são avaliados para promover os mesmos seqüenciamentos da solução guia. Buscas locais são realizadas sempre que o custo de S_p for menor que o custo de S_i e/ou menor que o custo de S_g , do mesmo modo como feito durante a primeira etapa.

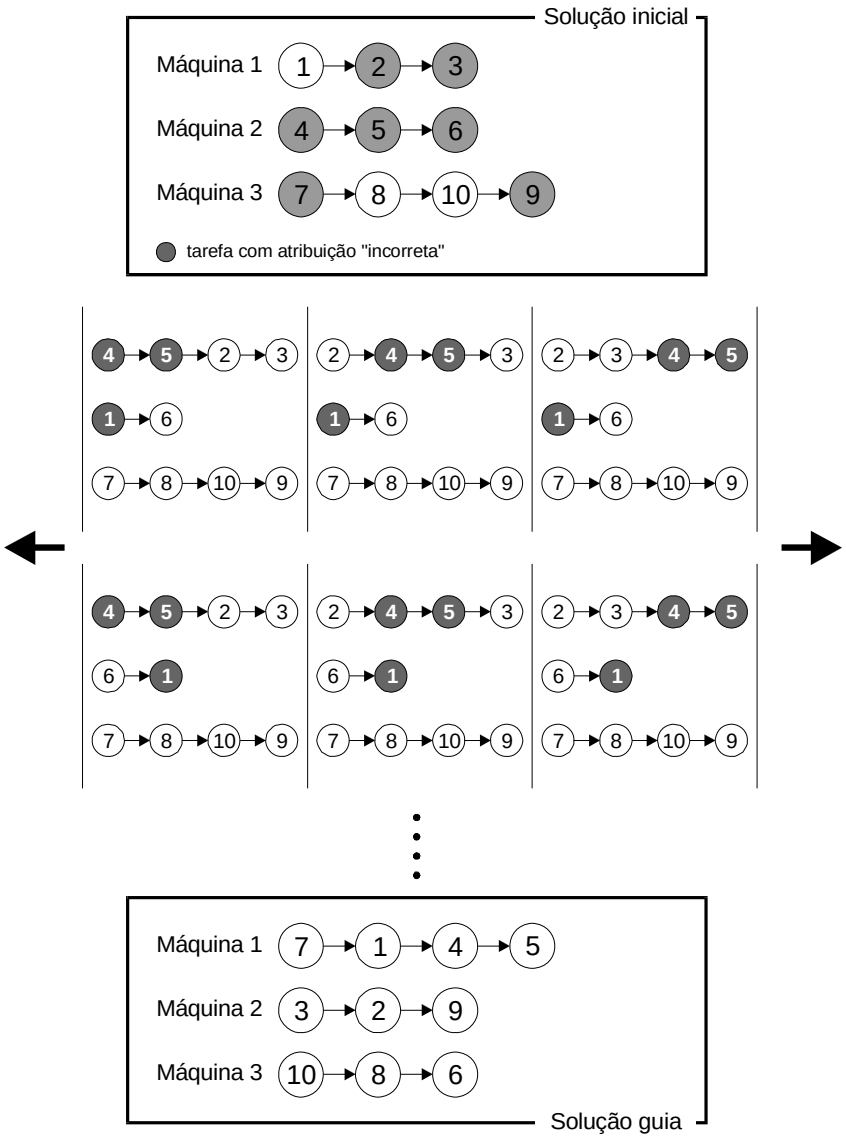

Figura 2.8 – Correção de atribuições na religação de caminho : Variante RC2

Resende e Ribeiro (2005) descrevem alternativas consideradas em recentes implementações de religação de caminhos entre duas soluções S_1 e S_2. São elas :

a) religação progressiva (*forward relinking*) : a solução inicial é a pior dentre S_1 e S_2;
b) religação regressiva (*backward relinking*) : a solução inicial é a melhor dentre S_1 e S_2;
c) religação regressiva e progressiva (*backward and forward relinking*) : duas trajetórias são geradas, começando, ora por S_1, ora por S_2.

2.4 Busca Tabu

Busca Tabu foi introduzida por Glover (1986) e desenvolvida em detalhe por Glover (1989, 1990) e por Glover e Laguna (1997), como uma estratégia de busca local baseada em memória. Com esta estratégia, que está essencialmente fundamentada na proibição de certos movimentos, consegue-se ir além da otimalidade local. A memória tem estruturas de curto e de longo prazo. A forma usual da memória de curto prazo consiste em registrar os atributos da solução que foram alterados no passado recente, armazenando-os em uma lista tabu. Soluções que sejam alcançáveis a partir da inclusão e/ou exclusão de atributos presentes na referida lista são proibidas. O objetivo é evitar ciclos de soluções visitadas e forçar a investigação de outras regiões do espaço de soluções. Já a memória de longo prazo contém uma história seletiva das soluções encontradas durante a busca, bem como de seus atributos.

2.4.1 Solução inicial

A solução inicial é obtida aplicando-se o índice ATCS (Lee *et al.*, 1997), de modo semelhante àquele adotado na obtenção das soluções iniciais do GRASP básico. A única diferença fica por conta do parâmetro α, na Equação 2.5, que é fixado em zero. Desse modo, a solução inicial é definida de modo estritamente guloso, pois em cada passo do procedimento de construção da solução inicial tem-se uma lista restrita, em geral de tamanho unitário, cujos elementos são as tarefas não alocadas com índice de prioridade igual ao máximo índice.

2.4.2 Busca local

Do mesmo modo como feito no GRASP, são definidas duas variantes de Busca Tabu. Na primeira realizam-se buscas locais alternadas nas vizinhanças de troca e de inserção, enquanto que na segunda a alternância se dá entre as vizinhanças *cross* e *Or-Opt*. A mudança de um tipo de vizinhança para outro ocorre após um número pré-determinado de iterações sem melhoria da solução tomada como partida para o tipo de vizinhança em questão. A solução de partida (S_0) para a vizinhança V_i pode ser a última solução (S_L) ou a solução de mínimo local (S_*) obtida com a vizinhança V_{i-1} (anterior). A escolha entre uma e outra depende da história da busca, como detalhado a seguir em uma transição da vizinhança de trocas internas para a vizinhança de trocas externas.

Seja Π_0^ϕ $(\phi = 1,2,...,m)$ a seqüência de tarefas atribuídas à máquina ϕ, na solução atual S_0. Em cada uma das m máquinas executam-se movimentos de trocas de pares de tarefas, enquanto houver melhoria local ou enquanto um número pré-determinado de movimentos sem melhoria em relação ao mínimo local não tiver sido alcançado. Sejam S_*^ϕ e S_L^ϕ a melhor e a última solução, respectivamente, encontradas ao se explorar a vizinhança de trocas internas na máquina ϕ. A solução a ser passada para a vizinhança de trocas de pares de tarefas entre máquinas será composta pelos mínimos locais de cada máquina, caso o custo de S_*^ϕ seja menor que o custo (inicial) de S_L^ϕ, para alguma máquina ϕ. Caso contrário, a vizinhança de trocas entre máquinas terá como solução de partida aquela composta pelas últimas soluções de cada máquina, ou seja, S_L^ϕ. A motivação é não interromper uma trajetória de descida, dando à vizinhança seguinte uma oportunidade de melhorar a solução ainda mais.

Na Figura 2.9 é mostrado um exemplo em que a exploração da vizinhança deve ser abandonada após 3 iterações sem melhoria local. O exemplo refere-se a uma instância com 2 máquinas. A linha cheia representa a trajetória da Busca Tabu (BT). A linha tracejada indica a solução de mínimo custo ao longo das iterações consideradas. Na máquina 1 observam-se melhorias da solução de partida (iteração 0) até a terceira iteração (mínimo local). A partir deste ponto ocorrem apenas movimentos de piora e a vizinhança de trocas internas na máquina 1 é abandonada ao final da sexta iteração. Já na máquina 2 não há movimentos de melhoria em

relação à solução da iteração 0. Dessa forma, a solução de partida para os movimentos de troca de pares de tarefas entre máquinas é composta por S_*^1 (correspondente à solução de custo mínimo – obtida na iteração 3) e S_*^2 (iteração 0).

Na Figura 2.10 tem-se uma situação em que, nas duas máquinas, as trocas internas realizadas não melhoram a solução da iteração 0. Em tal caso, a solução de partida para os movimentos de trocas externas é composta por S_3^1 e S_3^2, que correspondem às soluções da iteração 3 (trajetória "BT") nas máquinas 1 e 2, respectivamente.

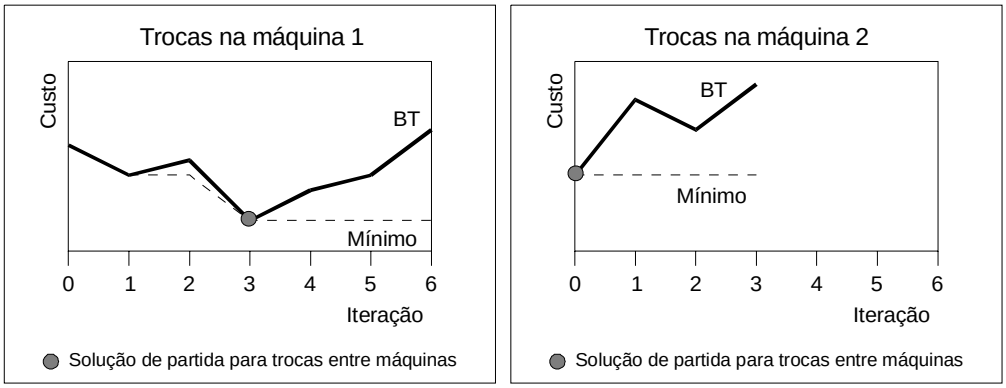

Figura 2.9 – Composição da solução de partida com mínimos locais

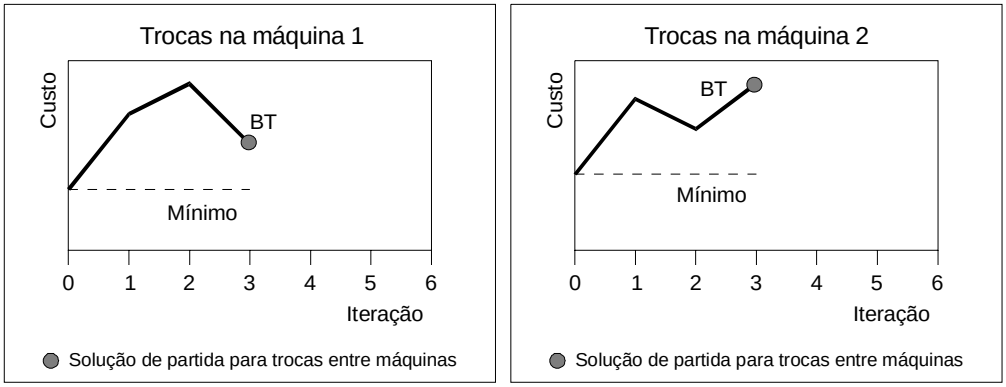

Figura 2.10 – Composição da solução de partida com últimos movimentos

2.4.3 Memória de curto prazo

A memória de curto prazo tem por objetivo proibir movimentos que levem a soluções visitadas em um passado recente. Ela é estabelecida em termos dos atributos que caracterizam as soluções. Em problemas de programação de tarefas com máquinas idênticas, os atributos são constituídos pelas relações de precedência entre tarefas (tanto as relações existentes quanto as inexistentes). Desse modo, os critérios de proibição de movimentos são estabelecidos em termos das relações de precedência que são criadas e/ou que são desfeitas, como exemplificado a seguir.

Seja $\Pi_x^{\phi} = \{\pi(1),...,\pi(i-1),\pi(i),\pi(i+1),...,\pi(j-1),\pi(j),\pi(j+1),...\}$ a seqüência de tarefas atribuídas à máquina ϕ, na iteração (x) atual. Nesta solução, $\pi(i)$ representa a tarefa que ocupa a i-ésima posição na seqüência de tal máquina. A troca entre as tarefas $\pi(i)$ e $\pi(j)$ faz surgir os arcos $ac_1 = (\pi(i-1),\pi(j))$, $ac_2 = (\pi(j),\pi(i+1))$, $ac_3 = (\pi(j-1),\pi(i))$ e $ac_4 = (\pi(i),\pi(j+1))$, enquanto que os arcos $ad_1 = (\pi(i-1),\pi(i))$, $ad_2 = (\pi(i),\pi(i+1))$, $ad_3 = (\pi(j-1),\pi(j))$ e $ad_4 = (\pi(j),\pi(j+1))$ deixam de existir.

Dois critérios de proibição, aqui denominados C_{in} e C_{out}, são considerados, de modo independente. Cada um deles dá origem a uma variante de Busca Tabu.

Critério de proibição C_{in}

Seja M_1 uma matriz n x n cujos elementos, $M_1(i,j)$, armazenam a iteração até a qual a criação de cada arco (ausente da solução atual) está proibida. De acordo com o critério C_{in}, o *status* do movimento é "proibido" se $M_1(i,j)$ for maior do que a iteração (x) atual, para qualquer par de tarefas (i,j), tal que (i,j) corresponda a um dos arcos a serem criados pelo movimento. No exemplo anterior, o *status* do movimento é "proibido" se:

$$\left(M_1(\pi(i-1),\pi(i))>x\right) \quad ou \quad \left(M_1(\pi(i),\pi(i+1))>x\right) \quad ou$$
$$\left(M_1(\pi(j-1),\pi(j))>x\right) \quad ou \quad \left(M_1(\pi(j),\pi(j+1))>x\right)$$

Laguna *et al.* (1995), Potvin *et al.* (1996) e Bilge *et al.* (2004) são exemplos de trabalhos nos quais o critério de proibição está definido em termos do retorno de arcos recentemente retirados da solução.

Critério de proibição C_{out}

Seja M_2 uma matriz n x n cujos elementos, $M_2(i,j)$, armazenam a iteração até a qual a retirada de cada arco (presente na solução atual) está proibida. De acordo com o critério de proibição C_{out}, o *status* do movimento é "proibido" se $M_2(i,j)$ for maior do que a iteração (x) atual, para qualquer par de tarefas (i,j), tal que (i,j) corresponda a um dos arcos a serem retirados. No exemplo anterior, o *status* do movimento é "proibido" se:

$$\left(M_1(\pi(i-1),\pi(j))>x\right) \quad ou \quad \left(M_1(\pi(j),\pi(i+1))>x\right) \quad ou$$
$$\left(M_1(\pi(j-1),\pi(i))>x\right) \quad ou \quad \left(M_1(\pi(i),\pi(j+1))>x\right)$$

O objetivo ao considerar os dois critérios de proibição descritos anteriormente é verificar o comportamento da Busca Tabu diante de critérios com significados completamente opostos, que apresentam diferentes graus de restritividade e exigem diferentes durações tabu.

Duração tabu

O número de iterações durante as quais um determinado atributo é considerado proibido é escolhido aleatoriamente, com probabilidade uniforme, no intervalo $\left[Min_{Tenure}\ ,\ Max_{Tenure}\right]$, em que os extremos Min_{Tenure} e Max_{Tenure} estão relacionados às dimensões da instância.

Para os movimentos internos às máquinas (troca, inserção e *Or-Opt*) o intervalo de escolha da duração tabu é $\left[Min_{Tenure} = a * n \div m,\ \ Max_{Tenure} = b * n \div m\right]$, em que a e b são números reais positivos sujeitos a uma calibração. A motivação para que os extremos sejam proporcionais a $n \div m$ está associada ao fato deste ser o número médio de arcos nas máquinas. Para os movimentos entre máquinas (troca, inserção e *cross*) o intervalo de escolha da duração tabu é $\left[Min_{Tenure} = a * n * m,\ \ Max_{Tenure} = b * n * m\right]$, com a e b números reais positivos que também estão sujeitos a uma calibração. Os extremos são proporcionais a $n * m$ pois quanto maiores os totais de tarefas e de máquinas, maiores as alternativas de movimentos que promovem alterações de atribuições.

Critério de aspiração

O critério de aspiração consiste em admitir um movimento tabu se ele permitir alcançar uma solução de menor custo que aquele da melhor solução encontrada durante a busca.

2.4.4 Memória de longo prazo

As estratégias de longo prazo são ativadas ao se perceber uma estagnação da Busca Tabu de curto prazo. Elas são classificadas em estratégias de diversificação, projetadas para levar a busca para regiões pouco exploradas e estratégias de intensificação, por meio das quais procura-se explorar, de modo mais detalhado, as regiões associadas às melhores soluções encontradas até então.

Quatro estratégias de memória de longo prazo são avaliadas, em quatro variantes distintas. Na primeira delas, denominada TDiv, a função de avaliação é alterada a partir de informações de freqüência de residência, nos moldes sugeridos por Glover e Laguna (1997). Na segunda, denominada TRC, executam-se religações de caminhos entre a solução atual e uma solução de elite. A terceira variante, TDivRC, é uma combinação das duas primeiras. Na quarta,

adaptada do trabalho de Rochat e Taillard (1995), a Busca Tabu é reinicializada a partir de uma solução criada por meio de uma composição que envolve um banco de seqüências de elite. Esta última é denominada TRT.

Freqüências de residência – Variante TDiv

Esta variante faz uso das freqüências de residência para promover uma diversificação da busca. Como sugerido por Glover e Laguna (1997), as regras de escolha são modificadas a fim de trazer para a solução aqueles atributos pouco presentes nas iterações anteriores. A modificação consiste no emprego de penalizações ao custo (Z) da solução, as quais dependem das medidas das freqüências de residência, como na Equação 2.15, em que *Penalty* é função da medida de freqüência e γ um parâmetro ajustável que controla o grau de diversificação desejado.

$$Z_{Modificado} = (1 + \gamma . Penalty)\, Z_{Original} \tag{2.15}$$

Considere o exemplo de um movimento *cross* mostrado na Figura 2.11.

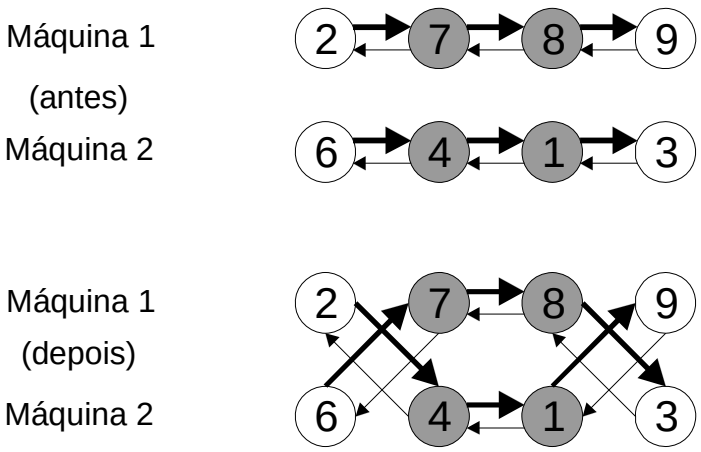

Figura 2.11 – Criação de arcos em um movimento *cross*

Supondo que não haja proibições ao movimento, a sua realização resultaria na criação dos arcos *(2,4), (6,7), (1,9) e (8,3)*. A alteração da regra de escolha, baseada em freqüências de residência consiste em penalizar o custo do movimento de acordo com a Equação 2.16.

$$Z_{Modificado} = \left(1 + \gamma \frac{\max(R(2,4), R(6,7), R(1,9), R(8,3))}{R_{\max}}\right) Z_{Original} \quad (2.16)$$

R(2,4), *R(6,7)*, *R(1,9)*, *R(8,3)* são as freqüências de residência dos arcos a serem criados pelo movimento, enquanto que R_{max} representa o maior valor de freqüência de residência para todo e qualquer par ordenado de tarefas (i,j) da instância. Quanto maior a máxima freqüência de residência dos arcos a serem criados por um dado movimento, maior a penalização a ele imposta. Assim, movimentos associados à criação de arcos pouco presentes nas soluções visitadas passam a ter maiores oportunidades.

O procedimento geral da variante **TDiv** é constituído pelos seguintes passos:

1. Realize movimentos segundo as regras de curto prazo, até que seja detectada a estagnação por um número x_1 de movimentos;
2. Realize x_2 movimentos de diversificação, baseados em freqüência de residência. Durante a diversificação, o custo da solução recebe uma penalização diretamente proporcional à máxima freqüência de residência dos arcos a serem nela inseridos. Seja S_{div} a solução obtida ao final dos x_2 movimentos de diversificação;
3. Desabilite a penalização por freqüência e, tomando a solução S_{div} como ponto de partida, volte para a busca local de curto prazo (passo 1).

Religações de caminhos intermediárias – Variante TRC

Nesta variante, religações de caminho intermediárias são empregadas como única estratégia de intensificação / diversificação, conforme procedimento geral descrito a seguir.

1. Realize movimentos segundo as regras de curto prazo, até que seja detectada a estagnação por um número x_1 de movimentos. Seja S_{x1} a solução obtida com o último movimento. Mantenha atualizado o conjunto de soluções de elite;
2. Ligue a solução atual, S_{x1}, à mais distante solução do conjunto de elite. Durante o processo de religação, realize buscas locais (apenas movimentos de melhoria), sempre que o custo da solução intermediária for o menor dentre todos os custos das soluções obtidas ao longo do caminho de religação;
3. Retome a Busca Tabu de curto prazo, a partir de S_{x1} (passo 1).

Freqüências de residência e Religações de caminhos – Variante TDivRC

Esta variante é formada pela estratégia de diversificação da variante **TDiv**, acrescida de religações de caminho intermediárias, como estratégia de intensificação / diversificação. O procedimento geral é descrito a seguir.

1. Realize movimentos segundo as regras de curto prazo, até que seja detectada a estagnação por um número x_1 de movimentos. Mantenha atualizado o conjunto de soluções de elite;
2. Realize x_2 movimentos de diversificação, baseados em freqüência de residência. Durante a diversificação, o custo da solução recebe uma penalização diretamente proporcional à máxima freqüência de residência dos arcos a serem nela inseridos. Seja S_{div} a solução obtida ao final dos x_2 movimentos de diversificação;
3. Desabilite a penalização por freqüência e, a partir da solução S_{div}, realize movimentos segundo as regras de curto prazo, até alcançar um mínimo local S_{dmin};
4. Ligue o mínimo local S_{dmin} à mais distante solução do conjunto de elite. Durante o processo de religação, realize buscas locais (apenas movimentos de melhoria), sempre que o custo da solução intermediária for o menor dentre todos os custos das soluções obtidas ao longo do caminho de religação;
5. Retome a Busca Tabu de curto prazo, a partir de S_{dmin} (passo 1).

Reinícios a partir de um banco de seqüências – Variante TRT

Em Rochat e Taillard (1995) é apresentado um procedimento para explorar a interação entre estratégias de diversificação e de intensificação, para o problema de roteamento de veículos. A diversificação está baseada no uso estratégico da aleatorização para as tomadas de decisão, ou seja, uma aleatorização guiada por uma função de probabilidade que favorece, fortemente, a escolha de variáveis mais bem avaliadas. Por sua vez, a essência da intensificação consiste na geração de soluções fazendo-se referência à noção de variáveis consistentes e fortemente determinadas. Uma variável fortemente determinada é aquela que não pode ter seu valor alterado, em uma solução de alta qualidade, sem provocar uma séria degradação na qualidade da solução, ou até mesmo torná-la infactível. Já uma variável consistente é aquela que freqüentemente assume um valor específico, ou uma faixa de valores muito estreita, em soluções de alta qualidade.

O procedimento de Rochat e Taillard (1995) é constituído por quatro fases. Na primeira, de inicialização, uma busca local, aplicada a distintas soluções de partida, leva à formação de um conjunto de $\boldsymbol{I}$ soluções distintas. Segundo Rochat e Taillard, o que se espera é que todas as informações necessárias para criar soluções de alta qualidade estejam presentes nas soluções do conjunto $\boldsymbol{I}$, embora de uma maneira não evidente. Assim sendo, a fase de inicialização cria um conjunto de rotas que inclui elementos não muito diferentes daquelas rotas presentes em boas soluções. Ainda segundo Rochat e Taillard, criar boas rotas é fácil. O desafio é definir um conjunto delas que forme uma solução factível. As $\boldsymbol{I}$ soluções distintas dão origem a um conjunto $\boldsymbol{T}$ de rotas.

Na segunda fase o objetivo é extrair um subconjunto de boas rotas do conjunto $\boldsymbol{T}$, a fim de formar uma nova solução. No processo de extração, dá-se preferência às rotas pertencentes às melhores soluções do conjunto $\boldsymbol{I}$, sem, contudo, excluir totalmente aquelas pertencentes às piores soluções. Tal conduta é motivada pelo fato de que se uma determinada solução contém rotas presentes em uma boa solução, então é provável que tal solução seja de melhor qualidade que uma outra que não contenha tais rotas. Para implementar a segunda fase, cada rota de $\boldsymbol{T}$ recebe um indicador com o valor da solução à qual ela pertence. Em seguida, o conjunto $\boldsymbol{T}$ é ordenado de acordo com valores crescentes do indicador. Rotas com apenas um

cliente são eliminadas. Inicia-se então o processo de extração de rotas, com maior probabilidade de escolha daquelas com menores indicadores e ignorando-se todas aquelas que incluam pelo menos um cliente atendido por alguma rota já extraída. O processo continua enquanto for possível extrair rotas do conjunto $\boldsymbol{T}$.

Seja S o conjunto de rotas extraídas na fase anterior. Eventualmente o conjunto S pode não contemplar todos os clientes. Neste caso, S é uma solução parcial. A terceira fase do procedimento consiste na obtenção de uma solução factível, a partir da solução parcial S. Para tanto, o conjunto de clientes não atendidos pela solução parcial pode ser tratado como um problema independente, a ser resolvido pelo mesmo procedimento de busca local utilizado na obtenção do conjunto $\boldsymbol{I}$ da primeira fase. Em seguida, as rotas da solução deste novo problema são adicionadas à solução parcial S, formando, assim, uma solução factível.

Na quarta e última fase, a solução S é tomada como ponto de partida para uma busca local, na tentativa de melhorá-la. Havendo a melhoria, as rotas da nova solução recebem o seu indicador (custo da solução) e são inseridas no conjunto $\boldsymbol{T}$, realimentando o processo, até que algum critério de parada seja satisfeito.

Deve-se notar que uma dada rota pode estar presente em mais de uma solução. Tal fato pode ser interpretado como uma medida de consistência da rota e para levá-lo em consideração, a rota é incluída múltiplas vezes no conjunto $\boldsymbol{T}$, uma para cada solução à qual pertença. Assim, uma rota consistente tem maior probabilidade de ser escolhida.

O procedimento proposto por Rochat e Taillard (1995) pode ser prontamente aplicado ao problema de programação de tarefas. Nesse caso, o conjunto $\boldsymbol{T}$ passa a ser um banco de seqüências de tarefas alocadas a uma mesma máquina. No presente trabalho, o procedimento de Rochat e Taillard (1995) é utilizado como estratégia de longo prazo para a Busca Tabu, conforme detalhado a seguir.

1. Realize movimentos segundo as regras de curto prazo, até que seja detectada a estagnação global por um número x_1 de movimentos. Mantenha atualizado o conjunto de soluções de elite;
2. Esvazie o conjunto $\boldsymbol{T}$, cujos elementos são seqüências de tarefas alocadas a uma mesma máquina;
3. Defina S como uma solução parcial (inicialmente vazia);
4. Para cada uma das soluções do conjunto de elite
   a) Calcule, para cada máquina ϕ, o custo médio por tarefa processada

   $$\bar{z}_{\phi} = \frac{1}{p}\sum_{j=1}^{p} t_j T_j \quad \forall\ j \in \phi$$

   b) Faça o indicador de cada uma das máquinas (Z_{ϕ}) igual ao custo da solução de elite;
   c) Introduza as máquinas, isto é, as seqüências de tarefas alocadas a uma mesma máquina, uma a uma, no conjunto $\boldsymbol{T}$;
5. Ordene os elementos do conjunto $\boldsymbol{T}$ em função de valores crescentes de seus indicadores (passo 4b). Havendo dois ou mais elementos com mesmo indicador, ordene-os em função de valores crescentes do custo médio por tarefa processada (passo 4a);
6. Faça *MáquinaDisponível = 1*;
7. Enquanto $|\boldsymbol{T}| > 0$ e *MáquinaDisponível* $\leq m$
   a) Para cada elemento $e \in \boldsymbol{T}$, calcule a probabilidade de sua extração, como: $P(e) = 2\left(|T|+1-e\right)/|T|/\left(|T|+1\right)$. Observe que ao melhor elemento de $\boldsymbol{T}$ (e = *1*) é associada uma probabilidade de extração igual a $P(1) = 2/\left(|T|+1\right)$, enquanto que ao pior ($e = |T|$), é associada a probabilidade $P(|T|) = 2/|T|/\left(|T|+1\right)$;
   b) Selecione um elemento do conjunto $\boldsymbol{T}$, com probabilidade definida no passo 7(a). Seja e* o elemento escolhido.
   c) Atribua a seqüência de tarefas de e* à *MáquinaDisponível*;
      Isto equivale a atualizar a solução parcial $\boldsymbol{S}$: $\boldsymbol{S} = \boldsymbol{S} \cup e$*
   d) Faça *MáquinaDisponível* = *MáquinaDisponível* + 1;
   e) Elimine, do conjunto $\boldsymbol{T}$, o elemento e*, bem como todos os outros elementos que contenham pelo menos uma tarefa presente no elemento e*.
8. Defina um conjunto $\overline{N}$ cujos elementos são as tarefas ausentes da solução parcial $\boldsymbol{S}$;

9. Enquanto $\left|\overline{N}\right| > 0$
   a) Denote por t^* o menor dentre todos os instantes de liberação das máquinas. Seja ϕ^* a máquina liberada mais cedo. Se houver mais de uma máquina liberada no instante t^*, uma delas é escolhida aleatoriamente, com probabilidade uniforme. Seja k a última tarefa alocada na máquina ϕ^*;
   b) Para cada tarefa j pertencente ao conjunto $\overline{N}$, calcule o índice de prioridade ATCS (Equação 2.1). Seja $I_{\max}$ o máximo índice de prioridade calculado;
   c) Seja j^* a tarefa com índice de prioridade $I_{\max}$. Se houver mais de uma tarefa com índice $I_{\max}$, escolha uma delas aleatoriamente;
   d) Retire a tarefa j^* do conjunto $\overline{N}$ e coloque-a na última posição da máquina ϕ^*.
10. Retorne ao passo 1, reiniciando a busca de curto prazo a partir da solução $\boldsymbol{S}$ recém-construída.

2.4.5 Pós-otimização via religações de caminhos

Como estratégia de pós-otimização, adotada para todas as variantes descritas anteriormente, empregam-se religações de caminho (regressiva e progressiva) entre cada uma das soluções de elite e a incumbente, atualizando-se o conjunto de elite, sempre que possível. O processo é aplicado enquanto houver atualizações no conjunto de elite.

Capítulo 3

Minimização da soma ponderada de custos de avanço e de atraso em máquinas paralelas não-relacionadas

3.1 Caracterização do problema e revisão bibliográfica

Neste problema procura-se definir a atribuição de um conjunto de n tarefas independentes a um conjunto de m máquinas paralelas, não-relacionadas e continuamente disponíveis, bem como a seqüência de processamento das tarefas em cada máquina e o instante de início de cada uma delas, visando a minimização da soma ponderada de custos de avanço e de atraso em relação às datas de entrega das tarefas.

Para brevidade em referências futuras, este problema é denominado **PSET** (das palavras-chave em inglês *Parallel machines*, *Setup times, Earliness* e *Tardiness*). Cada tarefa i, cujo processamento não pode ser interrompido uma vez iniciado, é caracterizada pelo seguinte conjunto de parâmetros: tempo de processamento, p_i^{ϕ}, dependente da máquina, data de entrega, d_i, instante de liberação, r_i, penalização por unidade de tempo de atraso, t_i, e penalização por unidade de tempo de avanço, e_i. A transição entre duas tarefas adjacentes, i e j, demanda um tempo de preparação, s_{ij}^{ϕ}, que depende da seqüência de processamento das tarefas, bem como da máquina ϕ em que as duas tarefas são processadas. A preparação para o processamento de uma tarefa pode ser iniciada antes mesmo de seu instante de liberação. Para o processamento da tarefa i na primeira posição da seqüência da máquina ϕ é necessário um tempo de preparação inicial, s_{0i}^{ϕ}. Como no problema PST, os tempos de preparação respeitam a desigualdade triangular.

O atraso da tarefa i é definido como $T_i = \max(0, C_i - d_i)$, enquanto que o avanço é definido por $E_i = \max(0, d_i - C_i)$. Nestas duas expressões, C_i é o instante de conclusão do processamento da tarefa i. A função objetivo a ser minimizada é, portanto, $Z = \sum_{i=1}^{n} (t_i T_i + e_i E_i)$.

O estudo de problemas de programação de tarefas envolvendo penalizações por avanço e por atraso é mais recente que aquele voltado para problemas em que o objetivo envolve uma função não-decrescente do instante de conclusão do processamento da tarefa (função objetivo regular), tais como tempo médio de fluxo, soma ponderada de atrasos e *makespan*. Para estes, custos mais elevados resultam apenas do adiamento na conclusão das tarefas. Entretanto, hoje em dia a tônica é penalizar também a conclusão das tarefas antes do instante em que elas são requeridas (filosofia *just-in-time*). Concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade de disponibilização antecipada de capital, necessidade de espaço para armazenamento e necessidade de outros recursos para manter e gerenciar o estoque.

Problemas envolvendo penalizações por avanço e por atraso podem ser classificados em duas principais categorias, de acordo com o tipo de especificação das datas de entrega. Tem-se então problemas em que a data de entrega é única para todas as tarefas (*common due date*) e aqueles em que a cada tarefa é atribuída uma data de entrega específica (*distinct due dates*).

Baker e Scuder (1990) e Gordon et al. (2002) apresentam extensas revisões relacionadas ao problema de common due date. Baker e Scuder (1990) estabelecem as condições que devem ser satisfeitas por uma solução ótima para este tipo de problema e que, se consideradas no algoritmo de solução, permitem uma significativa redução do esforço computacional:

a) inexistência de tempos ociosos;

b) a seqüência deve ser *V-shaped*, isto é, as tarefas adiantadas devem ser ordenadas por valores não-crescentes da razão entre tempo de processamento e penalização por avanço (p_i / e_i), enquanto as tarefas atrasadas devem ser ordenadas por valores não-decrescentes da razão entre tempo de processamento e penalização por atraso (p_i / t_i);

c) uma tarefa é concluída exatamente na data de entrega;

d) para a tarefa concluída exatamente na data de entrega, supondo que ela ocupe a *i*-ésima posição na seqüência, tem-se: $\sum_{j=1}^{i}\left(e_{\pi(j)} + t_{\pi(j)}\right) > \sum_{j=1}^{n} t_{\pi(j)}$, em que $\pi(j)$ é a tarefa que ocupa a *j*-ésima posição da seqüência.

Dentre os trabalhos em data de entrega única, não citados por Baker e Scuder (1990) nem por Gordon *et al*. (2002), encontram-se: Chen e Powell (1999), Mondal e Sen (2001), Bank e Werner (2001), Ventura *et al*. (2002), Cheng *et al*. (2002), Mondal (2002), Panwalkar e Liman (2002), Sun e Wang (2003), Feldman e Biskup (2003), Rabadi *et al*. (2004), Suriyaarachchi e Wirth (2004), Hino *et al*. (2005), Liao e Cheng (2007).

Já em problemas com distintas datas de entrega, as propriedades apontadas por Baker e Scuder (1990) não são necessariamente satisfeitas, tornando mais difícil a obtenção de soluções de alta qualidade. Para tais problemas é necessário estabelecer se tempos ociosos devem (podem) ou não ser admitidos. Existem situações em que a ociosidade das máquinas não é permitida por implicar em custos mais elevados que aqueles decorrentes da conclusão antecipada das tarefas. Por outro lado, como apontado por Kanet e Sridharan (2000) há situações em que pode ser desejável manter uma máquina ociosa, ainda que haja alguma tarefa em condições de ter seu processamento iniciado. É o caso, por exemplo, em que uma tarefa de mais alta prioridade, ou à qual esteja associada uma elevada penalização por atraso, ainda não está disponível para processamento, mas poderá ter sua conclusão seriamente atrasada se alguma outra tarefa disponível, mas de baixa prioridade, for processada antes, apenas para evitar a ociosidade da máquina. Isto é especialmente verdadeiro em ambientes nos quais não são admitidas interrupções nos processamentos das tarefas. Assim, diante da inexistência de restrições à ociosidade das máquinas, a inserção de tempos ociosos entre tarefas poderá produzir soluções de menores custos.

Kanet e Sridharan (2000) estabelecem três situações principais nas quais a inserção deliberada de tempos ociosos é vantajosa:

a) quando houver mais que uma máquina;

b) quando houver tarefas liberadas para processamento em instantes diferentes;

c) quando a função objetivo for não-regular, isto é, quando o valor da função objetivo puder diminuir com o aumento do instante de conclusão do processamento da tarefa.

No problema considerado no presente trabalho as três situações são observadas. Portanto, a inserção de tempos ociosos é altamente recomendada.

Levantamento de Kanet e Sridharan (2000) indica diversos trabalhos que abordam a questão da alocação temporal das tarefas, com inserção de tempos ociosos, considerando uma seqüência definida de tarefas. São eles: Sidney (1977), Lakshminarayan *et al*. (1978), Garey *et al*. (1988), Davis e Kanet (1993), Lee e Choi (1995), Szwarc e Mukhopadhyay (1995). Todos estes trabalhos operam sobre o conjunto completo de tarefas da seqüência, começando pela última delas. A estes se soma o trabalho de Mazzini e Armentano (2001).

Um importante ponto destacado por Kanet e Sridharan (2000) refere-se à necessidade de integrar as etapas de atribuição, seqüenciamento e alocação temporal das tarefas. Uma estratégia tentadora, mas não recomendável, consiste em ignorar a inserção de tempos ociosos, numa primeira etapa, com objetivo de diminuir o esforço computacional, e explorar o espaço de soluções à procura de uma boa atribuição associada a boas seqüências em cada máquina. Em seguida, uma vez identificada a melhor solução da etapa anterior, aplica-se a inserção de tempos ociosos apenas a tal solução. Como exemplificado por Kanet e Sridharan (2000), esta estratégia pode levar a resultados desastrosos.

Kanet e Sridharan (2000) apontam diversas linhas de pesquisa, relativamente aos problemas de programação de tarefas com tempos ociosos, nas quais vale à pena investir. Uma delas refere-se ao desenvolvimento de procedimentos mais eficientes para a inserção de tempos ociosos – procedimentos que possam operar sobre subconjuntos de tarefas da seqüência. No presente trabalho um procedimento com esta característica é apresentado.

A seguir são apresentados os trabalhos mais recentes, envolvendo penalizações por avanço e por atraso, que consideram distintas datas de entrega.

James e Buchanan (1997) desenvolvem uma Busca Tabu fundamentada na utilização de um espaço de solução binário para fazer distinção entre tarefas adiantadas e atrasadas e conseguem resolver o problema de minimização da soma ponderada de custos de avanço e de atraso de um conjunto de tarefas, de modo mais eficiente que aquele alcançado com a utilização do espaço de soluções baseado na seqüência física das tarefas. James e Buchanan (1998) desenvolvem uma Busca Tabu híbrida que trabalha com o espaço de solução binário

nas iterações iniciais e com o espaço físico-seqüencial nas iterações subseqüentes. Os autores conseguem resolver instâncias de grande porte, de modo ainda mais eficiente. O ambiente é de uma única máquina, com tempos ociosos. Todas as tarefas estão disponíveis no instante inicial do horizonte de planejamento. As metodologias empregadas nestes dois trabalhos são limitadas a situações nas quais não há tempos de preparação.

Kolahan e Liang (1998) consideram a programação de tarefas em uma única máquina, com tempos de preparação dependentes da seqüência, cujo objetivo é minimizar a soma ponderada de custos de avanço, de atraso e de variações dos tempos de processamento, quando estes últimos são diminuídos ou aumentados em relação aos tempos básicos. Todas as tarefas estão disponíveis no instante inicial do horizonte de planejamento. Tempos ociosos não são admitidos. A metodologia empregada é a Busca Tabu, com movimentos de trocas de pares de tarefas. O tamanho da vizinhança é controlado de modo adaptativo, em termos da evolução histórica da função objetivo. Nas iterações iniciais, quando as taxas de melhoria são mais elevadas, toda a vizinhança é investigada. À medida que a taxa de melhoria diminui, passa-se a investigar apenas os movimentos envolvendo tarefas que, na seqüência corrente, estejam próximas entre si.

Almeida e Centeno (1998) desenvolvem uma metaheurística híbrida que emprega Busca Tabu e *Simulated Annealing*, em fases alternadas, para resolver o problema de minimização da soma ponderada de custos de avanço e de atraso na programação de tarefas em uma única máquina. Todas as tarefas estão disponíveis no instante inicial do horizonte de planejamento. Não são considerados tempos de preparação. Também não são admitidos tempos ociosos. Os autores observam que a alternância proposta produz melhores resultados que aqueles obtidos com cada uma das metaheurísticas isoladamente.

Heady e Zhu (1998) desenvolvem uma heurística construtiva gulosa que assume características da regra EDD, que fornece soluções ótimas para o problema de uma única máquina, quando o espalhamento entre datas de entrega das tarefas é grande em comparação à soma do tempo de processamento com o de preparação, e de uma regra desenvolvida por Kanet (1981), que fornece soluções ótimas, também para o problema de uma única máquina, quando as datas de entrega convergem para uma data única e as penalizações por atraso e por avanço são iguais. O foco de Heady e Zhu (1998) é a minimização da soma ponderada de

custos de avanço e de atraso em máquinas paralelas não-relacionadas. Os tempos de preparação são dependentes da seqüência. Todas as tarefas estão disponíveis no início do horizonte de planejamento. A heurística construtiva é aplicada para cada uma das m máquinas, fornecendo m seqüências completas, isto é, com todas as n tarefas. Para cada tarefa j, e em cada máquina, um fator de custo é calculado. O fator de custo é igual à soma do custo direto da tarefa j, dos custos ponderados de avanço (se existirem) das duas tarefas anteriores à tarefa j e dos custos ponderados de atraso (se existirem) das duas tarefas posteriores à tarefa j. Em seguida, a tarefa que apresentar a maior diferença entre os valores máximo e mínimo do fator de custo é retirada de todas as máquinas, menos daquela na qual o fator de custo é mínimo. O posicionamento temporal, com inserção de tempos ociosos, é atualizado segundo Szwarc e Mukhopadhyay (1995) e o processo é repetido até que cada tarefa esteja associada a apenas uma das m máquinas. O procedimento é aplicado a instâncias pequenas (com até 20 tarefas e 2 máquinas).

Serifoglu e Ulusoy (1999) empregam duas variantes de algoritmos genéticos, com e sem operador de *crossover*, na minimização da soma ponderada de custos de avanço e de atraso em máquinas uniformes, com tempos de preparação dependentes da seqüência. Os tempos de processamento, os instantes de liberação e as datas de entrega são específicos para cada tarefa, enquanto que as penalizações por avanço e por atraso, embora distintas entre si, são comuns a todas as tarefas. A inserção de tempos ociosos é realizada segundo o procedimento de Garey *et al*. (1988).

Balakrishnan *et al.* (1999) apresentam um modelo matemático para a minimização da soma ponderada de custos de avanço e de atraso em um ambiente de máquinas uniformes, com distintas datas de entrega, distintos instantes de liberação das tarefas e distintas penalizações por avanço e atraso. Os tempos de preparação respeitam a desigualdade triangular e a preparação para uma tarefa pode começar antes mesmo de sua chegada ao chão de fábrica para processamento. O modelo apresenta uma menor quantidade de variáveis inteiras de decisão, quando comparado a outros modelos para este tipo de problema. Não obstante, sua aplicação fica restrita a instâncias de dimensões muito pequenas (10 tarefas e 3 máquinas).

Zhu e Heady (2000) desenvolvem um modelo matemático para um problema muito semelhante àquele tratado por Balakrishnan *et al.* (1999). A diferença fica por conta do fato das máquinas serem não-relacionadas. A aplicabilidade do modelo é, como a do anterior, restrita a instâncias de pequenas dimensões (9 tarefas e 3 máquinas).

Radhakrishnan e Ventura (2000) desenvolvem uma heurística construtiva e uma implementação de *Simulated Annealing* para a minimização da soma absoluta de avanços e atrasos na programação de tarefas em máquinas paralelas idênticas. Os tempos de preparação são dependentes da seqüência, enquanto que tempos ociosos não são admitidos. Todas as tarefas estão disponíveis no instante zero. Na heurística construtiva as tarefas são ordenadas de acordo com valores crescentes do instante mais cedo em que cada uma pode ser concluída, sem contudo estarem sujeitas à penalização de avanço. O ordenamento é dinâmico, posto que depende dos instantes em que as tarefas estão liberadas, bem como da última tarefa alocada no programa parcial. A cada passo executa-se a alocação da tarefa que puder ser concluída mais cedo, mas que não esteja adiantada. No *Simulated Annealing* apenas movimentos de trocas de pares de tarefas são admitidos. Dessa forma, os totais de tarefas atribuídas a cada máquina, na fase construtiva, não são alterados. O desempenho do *Simulated Annealing* é confrontado com soluções exatas, para instâncias com até 10 tarefas e 2 máquinas. Para instâncias maiores (até 80 tarefas e 15 máquinas) são apresentados os ganhos obtidos frente à heurística construtiva.

Mazzini e Armentano (2001) consideram a programação de tarefas em uma única máquina, com objetivo de minimizar a soma ponderada de custos de avanço e de atraso. Uma heurística construtiva é proposta para determinar a seqüência de processamento e simultaneamente inserir tempos ociosos. Em seguida, uma busca local com trocas de pares de tarefas adjacentes é aplicada. Tempos de preparação não são considerados. Nem todas as tarefas estão liberadas para processamento no instante inicial.

Wan e Yen (2002) aplicam um algoritmo de Busca Tabu de curto prazo ao problema de minimização da soma ponderada de custos de avanço e de atraso em relação a janelas de tempo para entrega, em lugar de datas de entrega específicas. O ambiente é de uma única máquina. Tempos de preparação não são considerados. Dada uma seqüência, a inserção de tempos ociosos é executada segundo procedimento especialmente desenvolvido para considerar janelas de tempo para entrega.

Ventura e Kim (2003) consideram o problema de programar um conjunto de tarefas de mesma duração, em um conjunto de máquinas paralelas idênticas, em que, além das máquinas, podem ser necessários recursos extras (de disponibilidade limitada ao longo do tempo, mas reutilizáveis). O objetivo é minimizar a soma dos desvios absolutos em relação às correspondentes datas de entrega, sujeito à disponibilidade dos recursos extras. Dois casos especiais são considerados. No primeiro, as tarefas demandam os recursos extras de modo específico (tanto em tipo quanto em quantidade). No segundo, existe apenas um tipo de recurso adicional e as tarefas ou não o consomem, ou o consomem em quantidade unitária. Cada tarefa está disponível em um instante próprio. Não são considerados tempos de preparação. O problema é formulado como um programa linear inteiro com variáveis binárias, sendo usada uma abordagem de relaxação Lagrangeana para obter limitantes inferiores de melhor qualidade.

Ventura e Radhakrishnan (2003) desenvolvem uma metodologia para restringir o horizonte de planejamento e para formular um modelo matemático para o problema de programação de tarefas em uma única máquina, com objetivo de minimizar a soma de custos de avanço e de atraso. Todas as tarefas estão disponíveis no instante inicial do horizonte de planejamento. Os tempos de preparação estão incluídos nos tempos de processamento. Relaxação lagrangeana sobre o conjunto de restrições de alocação temporal é usada para fornecer limitantes superiores de melhor qualidade (desvios de 3%) para problemas com até 100 tarefas. O procedimento de Garey *et al*. (1988) é empregado para determinar a alocação temporal das tarefas.

Sourd e Kedad-Sidhoum (2003) desenvolvem um algoritmo de *branch-and-bound* para definir a programação de um conjunto de tarefas em uma única máquina. As tarefas estão disponíveis no instante inicial do horizonte de planejamento e estão sujeitas a distintas penalizações de avanço e de atraso. Tempos de preparação não são considerados, bem como tempos ociosos não são admitidos. Problemas com até 30 tarefas são resolvidos.

Schaller (2004) desenvolve um algoritmo de *branch-and-bound* e uma busca local para a programação de tarefas em uma única máquina, com objetivo de minimizar a soma dos avanços e dos quadrados dos atrasos. Não são considerados tempos de preparação entre tarefas. Todas as tarefas estão disponíveis no instante inicial. A alocação temporal é obtida com a aplicação de um procedimento de inserção de tempos ociosos, desenvolvido especificamente

para a função objetivo do problema. Duas heurísticas são avaliadas na geração da solução de partida para a busca local. Na primeira, as tarefas são ordenadas segundo a regra EDD. Na outra, o algoritmo de *branch-and-bound* é aplicado para minimizar apenas a soma dos quadrados dos atrasos. Nos problemas resolvidos, o número de tarefas não passa de 25.

Valente e Alves (2005) tratam da minimização da soma ponderada de custos de avanço e de atraso em uma única máquina. Todas as tarefas estão disponíveis no instante zero e não há tempos de preparação. As penalizações por avanço e atraso são específicas para cada tarefa e podem ser distintas entre si. Tempos ociosos não são permitidos. São apresentadas duas regras de despacho. A primeira, denominada WPT_MS (das palavras em inglês *Weighted Processing Time and Mininum Slack*) é, segundo os autores, bastante similar à regra EXP-ET de Ow and Morton (1989). A função de prioridade leva em conta as penalizações de avanço e atraso, os tempos de processamento e as folgas das tarefas (data de entrega - tempo de processamento - instante de conclusão da última tarefa alocada). A segunda regra de despacho é um procedimento guloso e leva em conta o custo de alocação das tarefas nas duas próximas posições da seqüência parcial. As duas regras de despacho empregam fatores obtidos experimentalmente (fatores de *look ahead*) e que dependem das características específicas das instâncias. As características adotadas e o procedimento de definição dos fatores de *look ahead* são baseados no trabalho de Lee *et al*. (1997). Após a obtenção da solução inicial, uma série de movimentos de trocas de pares de tarefas é executada a fim de melhorar o custo total da solução. Primeiramente, são realizadas trocas de pares adjacentes, segundo a regra de dominância de Ow and Morton (1989). Em seguida, pares de tarefas não adjacentes são trocados, segundo a regra de dominância de Liaw (1999). Ciclos de trocas são executados até não mais ser possível melhorar a solução.

Hassin e Shani (2005) desenvolvem estudos teóricos a respeito de propriedades observadas em seqüências ótimas associadas a problemas de programação de tarefas em uma única máquina, nos quais, além das penalizações por avanço e por atraso, existe a possibilidade de não executar uma tarefa, à custa de uma pesada penalização. Nos problemas considerados não há tempos de preparação entre tarefas e todas elas estão disponíveis no instante zero. Os autores também generalizam o procedimento de alocação temporal de Garey *et al*. (1988) para o caso em que a função objetivo consiste na minimização da soma ponderada de custos de avanço e de atraso.

Tsai (2007) apresenta um algoritmo genético para o problema de minimização da soma ponderada de custos de avanço e de atraso. O ambiente considerado é de uma única máquina. As tarefas têm distintos instantes de liberação e estão sujeitas a distintas penalizações por avanço e por atraso. Tempos ociosos são permitidos. Não considera tempos de preparação entre tarefas.

3.2 Alocação temporal com inserção de tempos ociosos

A completa caracterização de uma solução consiste em definir as tarefas que serão processadas em cada máquina, a seqüência em que se dá o processamento e a alocação temporal de cada tarefa (definição do instante de início do processamento). A alocação temporal, uma vez definidas as atribuições e os seqüenciamentos, é efetuada em duas etapas. Primeiramente, todas as tarefas são alocadas o mais cedo possível, minimizando os custos de atraso e, ao mesmo tempo, produzindo a maximização dos custos de avanço. Em seguida, tempos ociosos são inseridos desde que o bastante provável acréscimo nos custos de atraso seja inferior à possível redução nos custos de avanço. A primeira etapa da alocação temporal das tarefas envolve um esforço computacional cuja complexidade é de ordem $n \div m$, enquanto que a segunda etapa, correspondente à inserção de tempos ociosos, tem complexidade $(n \div m)^2$. Ao separar o procedimento de alocação temporal em duas etapas, como aqui sugerido, pode-se aplicar a segunda de forma seletiva, apenas para aquelas situações em que o mínimo custo de atraso de uma solução candidata é inferior ao custo total da solução de referência atual, como esquematizado na Figura 3.1 a seguir.

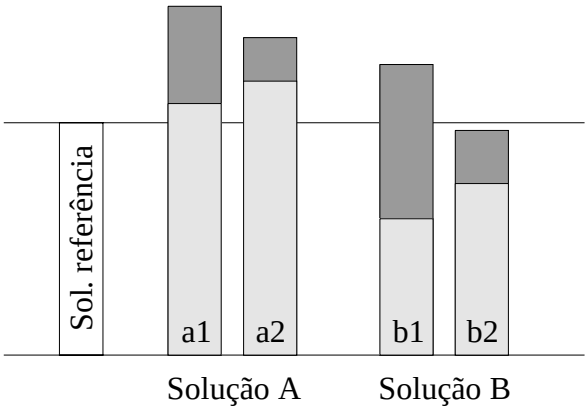

Figura 3.1– Inserção seletiva de tempos ociosos

Na Figura 3.1, seja S_R a solução de referência atual – pode ser a solução cuja vizinhança está sendo explorada ou a melhor vizinha já encontrada. Sobre a solução S_A, gerada posteriormente a S_R, aplica-se o procedimento de minimização de atrasos, que conduz ao custo total da coluna *a1*. Nesta coluna, o custo de atraso está indicado em cinza claro e o de avanço em cinza escuro. Observe que o custo total de atraso da solução S_A é maior que o custo de S_R. Isto significa que, mesmo com a aplicação do procedimento de inserção de tempos ociosos, o custo da solução S_A (coluna *a2*) continua maior que o da solução de referência. No caso da solução S_B, o mínimo custo total de atraso (mostrado em cinza claro na coluna *b1*) é menor que o custo total da solução de referência. Portanto, a inserção de intervalos ociosos pode revelar uma solução de menor custo que a de referência. Se este for o caso, a solução de referência é atualizada.

Seja $\Pi^{\phi} = \{\pi(1),...,\pi(i-1),\pi(i),\pi(i+1),...\}$ a seqüência de tarefas alocadas à máquina ϕ. $\pi(i)$ representa a tarefa que ocupa a i-ésima posição na seqüência. O procedimento de alocação temporal das tarefas, aplicável também a ambientes nos quais o instante de liberação da máquina é diferente de zero, é explicado a seguir.

Minimização dos custos de atraso (com maximização dos custos de avanço)

As alocações temporais com vistas à minimização dos custos de atraso começam pela primeira tarefa da seqüência, $\pi(1)$, cuja conclusão é dada em termos do instante de liberação da tarefa, de seu tempo de processamento, do instante de liberação da máquina, LM^{ϕ}, e do tempo de preparação inicial, $s^{\phi}_{0,\pi(1)}$, como indicado na Equação 3.1.

$$C_{\pi(1)} = \max(\, r_{\pi(1)}, LM^{\phi} + s^{\phi}_{0,\pi(1)}) + p^{\phi}_{\pi(1)} \qquad (3.1)$$

Cada uma das demais tarefas, a partir da segunda, tem sua conclusão dada pela Equação 3.2 (equação recursiva) que considera o instante de liberação e o tempo de processamento da tarefa em questão, o término da anterior e o tempo de preparação necessário para efetuar a transição entre as duas tarefas adjacentes.

$$C_{\pi(i)} = \max(r_{\pi(i)} \ , \quad C_{\pi(i-1)} + s^{\phi}_{\pi(i-1),\pi(i)}) + p^{\phi}_{\pi(i)} \tag{3.2}$$

O exemplo da Figura 3.2, cujos valores foram extraídos de uma instância com 60 tarefas e 6 máquinas, mostra em escala, na parte superior, o posicionamento mais cedo possível do conjunto de tarefas atribuídas a uma dada máquina. Também são indicadas as datas de entrega (d_i) e os instantes de liberação (r_i) das tarefas. Os quadros brancos representam os tempos de preparação entre tarefas. Os quadros cinza claro representam as tarefas adiantadas. Duas delas, as tarefas 1 e 3, têm seus inícios limitados pelos instantes de liberação. Os quadros cinza escuro e com numeração em branco representam as tarefas atrasadas. Na parte inferior tem-se o posicionamento após a inserção de tempos ociosos, cujo procedimento é explicado a seguir. Observa-se que as tarefas 1, 2 e 6, quadros na cor cinza médio, são concluídas nas respectivas datas de entrega.

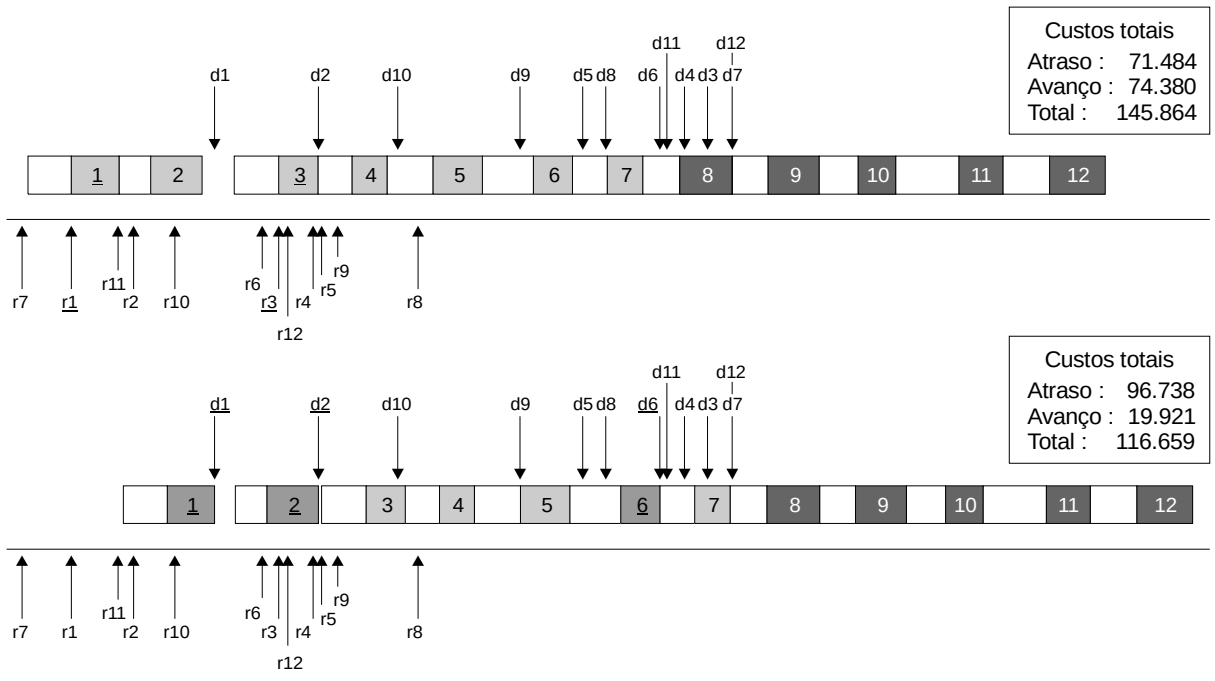

Figura 3.2 – Alocação temporal mais cedo possível (seqüência superior) *versus* Alocação temporal com inserção de tempos ociosos (seqüência inferior)

Inserção de tempos ociosos

A segunda etapa do posicionamento temporal também começa pela primeira tarefa da seqüência. Se ao final da primeira etapa a primeira tarefa estiver atrasada, então ela não sofrerá nenhum deslocamento para a direita, uma vez que isto aumentaria o custo de atraso a ela associado. Por outro lado, a primeira tarefa pode sofrer um deslocamento φ para a direita, de acordo com o estabelecido pela Equação 3.3, caso ela esteja adiantada.

$$\varphi = \min\left((d_{\pi(1)} - C_{\pi(1)}), (C_{\pi(2)} - p^{\phi}_{\pi(2)} - s^{\phi}_{\pi(1),\pi(2)} - C_{\pi(1)})\right) \tag{3.3}$$

Se o deslocamento imposto for dado por $\varphi = d_{\pi(1)} - C_{\pi(1)}$, a primeira tarefa alcança seu posicionamento ótimo, com seu instante de conclusão igual à sua data de entrega. Novos deslocamentos à direita tornam-se inviáveis, pois isto provocaria o surgimento de custo de atraso. Efetua-se, então, uma análise idêntica para a segunda tarefa.

Se o deslocamento da primeira tarefa for igual a $\varphi = C_{\pi(2)} - p^{\phi}_{\pi(2)} - s^{\phi}_{\pi(1),\pi(2)} - C_{\pi(1)}$, a primeira e a segunda tarefas passam a formar um bloco (conjunto de tarefas sem a existência de tempos ociosos entre elas). Neste caso, efetua-se uma análise em termos do custo marginal do bloco formado. O custo marginal (CM) é definido a partir da soma entre as penalizações por unidade de avanço (e_j) e por unidade de atraso (t_j) das tarefas do bloco. A soma leva em conta a condição de cada tarefa, como indicado na Equação 3.4.

$$\mathrm{CM} = \sum_{j \in atrasadas} t_j + \sum_{k \in concluídas\ na\ data\ de\ entrega} t_k - \sum_{i \in adiantadas} e_i \tag{3.4}$$

Considere que as duas tarefas estejam adiantadas. Neste caso, o custo marginal do bloco será negativo e igual, em módulo, à soma das penalizações por unidade de avanço de cada tarefa, ou seja, $(-e_{\pi(1)} - e_{\pi(2)})$. Custo marginal negativo indica que atrasar o bloco de tarefas é vantajoso. O deslocamento conjunto das duas tarefas é dado pela Equação 3.5.

$$\varphi = \min\left((d_{\pi(1)} - C_{\pi(1)}), (d_{\pi(2)} - C_{\pi(2)}), (C_{\pi(3)} - p^{\phi}_{\pi(3)} - s^{\phi}_{\pi(2),\pi(3)} - C_{\pi(2)})\right) \tag{3.5}$$

Se o deslocamento imposto ao bloco for $\varphi = (d_{\pi(1)} - C_{\pi(1)})$, a primeira tarefa alcança seu posicionamento ótimo, enquanto que a segunda tarefa continua adiantada. A primeira tarefa é retirada do bloco e a análise prossegue no sentido de definir o novo deslocamento a ser imposto à segunda tarefa.

Se o deslocamento imposto ao bloco for $\varphi = (d_{\pi(2)} - C_{\pi(2)})$, a segunda tarefa passa a ser concluída na sua data de entrega, enquanto que a primeira continua adiantada. Novos deslocamentos à direita somente serão viáveis se o custo marginal do bloco, agora dado por $(+t_{\pi(2)} - e_{\pi(1)})$, for negativo. Do contrário, o ganho obtido ao diminuir ainda mais o adiantamento da primeira tarefa não compensará a perda decorrente do atraso a ser imposto à segunda tarefa. Portanto, se o custo marginal do bloco for positivo, o bloco formado pelas duas primeiras tarefas terá alcançado seu posicionamento ótimo. Passa-se, então, a tratar a terceira tarefa.

Por fim, se $\varphi = (C_{\pi(3)} - p^{\phi}_{\pi(3)} - s^{\phi}_{\pi(2),\pi(3)} - C_{\pi(2)})$, a terceira tarefa é acrescentada ao bloco formado anteriormente pelas duas primeiras tarefas, que continuam adiantadas. Se a terceira tarefa estiver atrasada, novos deslocamentos à direita somente serão viáveis se o custo marginal do novo bloco, dado por $(+t_{\pi(3)} - e_{\pi(1)} - e_{\pi(2)})$, for negativo. Do contrário, o ganho obtido ao se aproximar ainda mais as duas primeiras tarefas das suas respectivas datas de entrega será menor que o aumento de custo de atraso da terceira tarefa. A análise para inserção de tempos ociosos prossegue ao longo da seqüência, até que seja posicionada a última tarefa.

O procedimento geral para inserção de tempos ociosos é mostrado a seguir.

1. Defina o bloco atual, B_p, como sendo um conjunto de tarefas consecutivas $\{\pi(p), \pi(p+1), ..., \pi(q)\}$, tal que $C_{\pi(i)} = C_{\pi(i+1)} - p^{\phi}_{\pi(i+1)} - s^{\phi}_{\pi(i),\pi(i+1)}$, com $i = p+1$ até q.

2. Elimine de B_p todas as tarefas atrasadas ou concluídas nas respectivas datas de entrega e que estejam posicionadas à esquerda da primeira tarefa adiantada. Em outras palavras, o bloco atual sempre começa com uma tarefa adiantada.

3. Calcule o custo marginal de B_p: $\mathrm{CM} = \sum_{j \in atrasadas} t_j + \sum_{k \in concluídas\ na\ data\ de\ entrega} t_k - \sum_{i \in adiantadas} e_i$.

4. Se CM < *0*, calcule o máximo deslocamento a ser imposto às tarefas de B_p: $\varphi = \min\left((d_{\pi(i^*)} - C_{\pi(i^*)}), (C_{\pi(q+1)} - p^{\phi}_{\pi(q+1)} - s^{\phi}_{\pi(q),\pi(q+1)} - C_{\pi(q)})\right)$, em que $(d_{\pi(i^*)} - C_{\pi(i^*)})$ representa o mínimo $(d_{\pi(i)} - C_{\pi(i)})$ avaliado apenas entre as tarefas adiantadas de B_p .

   5. Se $\varphi = (d_{\pi(i^*)} - C_{\pi(i^*)})$, após o deslocamento a tarefa $\pi(i^*)$ passa à condição de concluída na data de entrega. Se $\pi(i^*)$ for a primeira tarefa do bloco, retire-a de B_p. Volte para 3.

   6. Se $\varphi = (C_{\pi(q+1)} - p^{\phi}_{\pi(q+1)} - s^{\phi}_{\pi(q),\pi(q+1)} - C_{\pi(q)})$, após o deslocamento, anexe ao bloco atual B_p, todas as tarefas do bloco à direita de B_p e volte ao passo 3.

7. Se CM ≥ 0
   8. Enquanto CM ≥ 0 e o bloco B_p não estiver vazio, retire a primeira tarefa do bloco e recalcule o custo marginal. Volte ao passo 2 se o bloco B_p não for vazio ou se existir um próximo bloco. Caso contrário, o processo está terminado.

A seguir exemplifica-se a aplicação do procedimento de inserção de tempos ociosos. Os dados são os mesmos utilizados para a construção da Figura 3.2, que mostra o posicionamento mais cedo e o posicionamento final, após a inserção dos tempos ociosos.

Inicialmente todas as tarefas são posicionadas o mais cedo possível, respeitando-se os instantes de liberação. O custo total da solução é 145.864, sendo 74.380 por avanços (tarefas 1 a 7) e 71.484 por atrasos (tarefas 8 a 12). Isto é mostrado na Tabela 3.1.

Tabela 3.1 – Alocação temporal : Posicionamento mais cedo de todas as tarefas

Tarefa	r_j	s^{ϕ}_{ij}	*Início*	p_j	C_j	d_j	C_j - d_j	t_j	e_j	*Custo de atraso*	*Custo de avanço*
1	103	69	103	75	178	329	-151	12	35	0	5285
2	201	50	228	81	309	493	-184	99	91	0	16744
3	430	70	430	62	492	1107	-615	29	39	0	23985
4	484	54	546	55	601	1071	-470	44	1	0	470
5	498	73	674	78	752	910	-158	79	37	0	5846
6	404	80	832	62	894	1032	-138	61	76	0	10488
7	25	55	949	56	1005	1146	-141	94	82	0	11562
8	650	58	1063	83	1146	946	200	71	95	14200	0
9	523	57	1203	80	1283	811	472	84	18	39648	0
10	266	62	1345	59	1404	618	786	10	59	7860	0
11	176	100	1504	69	1573	1043	530	14	36	7420	0
12	445	74	1647	88	1735	1146	589	4	64	2356	0
										71484	74380

O bloco corrente é formado pelas tarefas 1 e 2. Ambas estão adiantadas. Portanto, o custo marginal do bloco é CM = – ($e_{\pi(1)}$ + $e_{\pi(2)}$) = – (35+91) = – 126. O deslocamento para a direita, a ser aplicado ao bloco, é o mínimo entre $d_{\pi(1)} - C_{\pi(1)}$, $d_{\pi(2)} - C_{\pi(2)}$ e $C_{\pi(3)} - p^{\phi}_{\pi(3)} - s^{\phi}_{\pi(2),\pi(3)} - C_{\pi(2)}$. Neste caso, o deslocamento é de 51 unidades de tempo, sendo limitado pelo início da preparação para a tarefa 3.

Após o deslocamento, situação indicada na Tabela 3.2, as tarefas 1 e 2 continuam adiantadas. O custo total da solução passa a ser 139.438 (67.954 de avanço e 71.484 de atraso). Agora o bloco passa a ser composto por todas as 12 tarefas. O custo marginal do bloco corresponde à soma das penalizações de atraso das tarefas 8 a 12, diminuída da soma das penalizações de avanço das tarefas 1 a 7. Isto é, CM = 183 – 361 = –178. Significa que é viável deslocar todo o bloco para a direita. O deslocamento deve ser o mínimo $d_{\pi(i)} - C_{\pi(i)}$, avaliado dentre as tarefas adiantadas. De acordo com os dados apresentados na Tabela 3.2, o deslocamento de todas as tarefas será de 100 unidades de tempo. Com isso, a tarefa 1 passa a ser concluída na sua data de entrega, como mostrado na Tabela 3.3.

Tabela 3.2 – Alocação temporal: Solução intermediária I

Tarefa	r_j	s^{ϕ}_{ij}	*Início*	p_j	C_j	d_j	C_j - d_j	t_j	e_j	*Custo de atraso*	*Custo de avanço*
1	103	69	154	75	229	329	-100	12	35	0	3500
2	201	50	279	81	360	493	-133	99	91	0	12103
3	430	70	430	62	492	1107	-615	29	39	0	23985
4	484	54	546	55	601	1071	-470	44	1	0	470
5	498	73	674	78	752	910	-158	79	37	0	5846
6	404	80	832	62	894	1032	-138	61	76	0	10488
7	25	55	949	56	1005	1146	-141	94	82	0	11562
8	650	58	1063	83	1146	946	200	71	95	14200	0
9	523	57	1203	80	1283	811	472	84	18	39648	0
10	266	62	1345	59	1404	618	786	10	59	7860	0
11	176	100	1504	69	1573	1043	530	14	36	7420	0
12	445	74	1647	88	1735	1146	589	4	64	2356	0
										71484	67954

Na situação indicada na Tabela 3.3 o custo total é de 121.638 (31.854 de avanço e 89.784 de atraso). A tarefa 1 é concluída na sua data de entrega, devendo, portanto, ser retirada do bloco. A análise da viabilidade de se impor novos deslocamentos (adiamentos) deve ser feita excluindo-se a tarefa 1, posto que ela já está no seu posicionamento ótimo. O custo marginal do bloco agora iniciado pela tarefa 2 é calculado como a soma das penalizações de atraso das tarefas 8 a 12, diminuída da soma das penalizações de avanço das tarefas 2 a 7,

continua negativo (CM = 183 – 326 = –143). Portanto, continua viável impor novos adiamentos ao bloco de tarefas. O deslocamento é de 33 unidades de tempo, provocando a conclusão da tarefa 2 na sua data de entrega, como indicado na Tabela 3.4.

Tabela 3.3 – Alocação temporal: Solução intermediária II

Tarefa	r_j	s_{ij}^{ϕ}	*Início*	p_j	C_j	d_j	C_j - d_j	t_j	e_j	*Custo de atraso*	*Custo de avanço*
1	103	69	254	75	329	329	0	12	35	0	0
2	201	50	379	81	460	493	-33	99	91	0	3003
3	430	70	530	62	592	1107	-515	29	39	0	20085
4	484	54	646	55	701	1071	-370	44	1	0	370
5	498	73	774	78	852	910	-58	79	37	0	2146
6	404	80	932	62	994	1032	-38	61	76	0	2888
7	25	55	1049	56	1105	1146	-41	94	82	0	3362
8	650	58	1163	83	1246	946	300	71	95	21300	0
9	523	57	1303	80	1383	811	572	84	18	48048	0
10	266	62	1445	59	1504	618	886	10	59	8860	0
11	176	100	1604	69	1673	1043	630	14	36	8820	0
12	445	74	1747	88	1835	1146	689	4	64	2756	0
										89784	31854

Como mostrado na Tabela 3.4 o custo total é 116.919 (21.096 de avanço e 95.823 de atraso). Com a retirada da tarefa 2, o bloco passa a ser iniciado pela tarefa 3. O custo marginal ainda é negativo (CM = 183 – 235 = – 52). O novo deslocamento (5 unidades de tempo) é ditado pela data de entrega da tarefa 6. Isto pode ser visto na Tabela 3.5.

Tabela 3.4 – Alocação temporal: Solução intermediária III

Tarefa	r_j	s_{ij}^{ϕ}	*Início*	p_j	C_j	d_j	C_j - d_j	t_j	e_j	*Custo de atraso*	*Custo de avanço*
1	103	69	254	75	329	329	0	12	35	0	0
2	201	50	412	81	493	493	0	99	91	0	0
3	430	70	563	62	625	1107	-482	29	39	0	18798
4	484	54	679	55	734	1071	-337	44	1	0	337
5	498	73	807	78	885	910	-25	79	37	0	925
6	404	80	965	62	1027	1032	-5	61	76	0	380
7	25	55	1082	56	1138	1146	-8	94	82	0	656
8	650	58	1196	83	1279	946	333	71	95	23643	0
9	523	57	1336	80	1416	811	605	84	18	50820	0
10	266	62	1478	59	1537	618	919	10	59	9190	0
11	176	100	1637	69	1706	1043	663	14	36	9282	0
12	445	74	1780	88	1868	1146	722	4	64	2888	0
										95823	21096

Na Tabela 3.5 é mostrado o posicionamento final das tarefas. O custo total é de 116.659 (19.921 de avanço e 96.738 de atraso). Entre a conclusão da tarefa 2 (instante 493) e o início da preparação para a tarefa 3 (instante 568 – 70 = 498) há um intervalo ocioso de 5 unidades de tempo.

Tabela 3.5 – Alocação temporal: Posicionamento final

Tarefa	r_j	s_{ij}^{ϕ}	*Início*	p_j	C_j	d_j	C_j - d_j	t_j	e_j	*Custo de atraso*	*Custo de avanço*
1	103	69	254	75	329	329	0	12	35	0	0
2	201	50	412	81	493	493	0	99	91	0	0
3	430	70	568	62	630	1107	-477	29	39	0	18603
4	484	54	684	55	739	1071	-332	44	1	0	332
5	498	73	812	78	890	910	-20	79	37	0	740
6	404	80	970	62	1032	1032	0	61	76	0	0
7	25	55	1087	56	1143	1146	-3	94	82	0	246
8	650	58	1201	83	1284	946	338	71	95	23998	0
9	523	57	1341	80	1421	811	610	84	18	51240	0
10	266	62	1483	59	1542	618	924	10	59	9240	0
11	176	100	1642	69	1711	1043	668	14	36	9352	0
12	445	74	1785	88	1873	1146	727	4	64	2908	0
										96738	19921

Com a tarefa 6 concluída na sua data de entrega, o custo marginal do bloco passa a ser dado pela soma das penalizações de atraso das tarefas 6, 8, 9, 10, 11 e 12, diminuída da soma das penalizações de avanço das tarefas 3, 4, 5 e 7 (CM = 244 – 159 = 85). O custo marginal é positivo. Portanto, qualquer novo deslocamento do bloco produz aumentos no custo total de atraso a uma taxa maior do que a queda no custo total de avanços. De acordo com o passo 8 do procedimento anteriormente apresentado, deve-se retirar a primeira tarefa do bloco e recalcular o custo marginal. Este passo deve ser repetido enquanto o custo marginal se mantiver positivo e enquanto ainda houver tarefas no bloco. Neste exemplo, todas as tarefas são retiradas. Como não há outro bloco posterior à tarefa 12, o procedimento de inserção de tempos ociosos está concluído.

A evolução dos custos (total, de atraso e de avanço) para este exemplo, a cada deslocamento realizado, é mostrada na Figura 3.3. É importante ressaltar que entre quaisquer dois deslocamentos à direita, sempre se tem posicionamentos factíveis de todas as tarefas. Desse modo, o procedimento de inserção de tempos ociosos pode ser interrompido a qualquer

momento, tão logo o custo de atraso de alguma solução intermediária ultrapasse o custo total de uma solução de referência (um limitante superior). No exemplo presente, se o custo total de referência fosse 90.000, nenhum deslocamento posterior ao terceiro precisaria ter sido realizado.

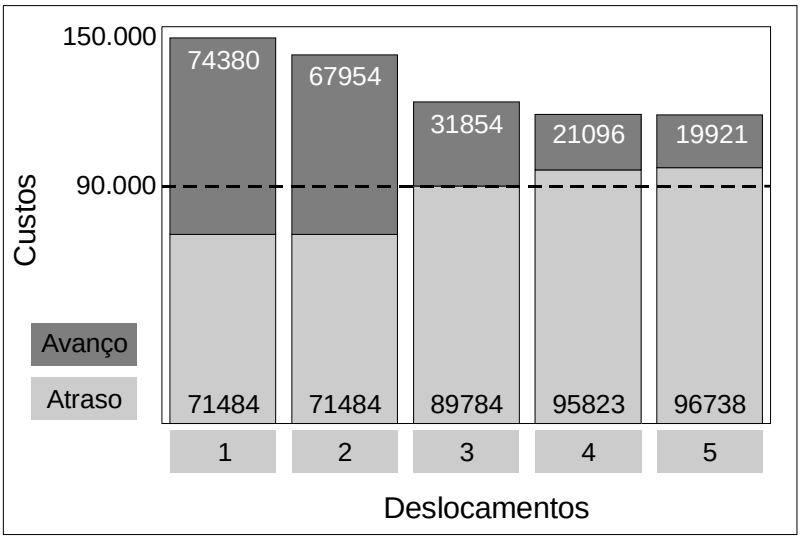

Figura 3.3 – Evolução dos custos durante inserção de tempos ociosos

3.3 GRASP

3.3.1 Função gulosa

Para o problema de minimização da soma ponderada de custos de avanço e de atraso são avaliadas funções gulosas baseadas no índice de prioridade ATCS, *Apparent Tardiness Cost with Setups*, (Lee *et al.*, 1997) e em algumas modificações impostas a este índice. As modificações procuram levar em consideração as penalizações por avanço e os instantes de liberação das tarefas.

Do mesmo modo como estabelecido no Capítulo 2, seja t^{ϕ} o instante de liberação da máquina ϕ, isto é, o instante de conclusão da última tarefa alocada em ϕ. Seja t^* o menor dentre todos os instantes de liberação e seja ϕ^* a máquina liberada mais cedo. Se houver mais de uma máquina liberada no instante t^*, uma delas é escolhida aleatoriamente, com probabilidade uniforme. Seja k a última tarefa alocada na máquina ϕ^*. Seja j uma tarefa pertencente ao conjunto de tarefas não alocadas, $\overline{N}$. Cada tarefa j é caracterizada por:

- tempo de processamento: p_j^{ϕ}
- custo por unidade de tempo de atraso: t_j
- custo por unidade de tempo de avanço: e_j
- instante de liberação da tarefa: r_j
- data de entrega: d_j
- tempo de preparação para execução imediatamente após a tarefa k: s_{kj}^{ϕ}

Além disso, considere que $\overline{p}$ e $\overline{s}$ representam o tempo de processamento médio e o tempo de preparação médio das tarefas não alocadas. No cálculo de $\overline{s}$ são levados em conta apenas os tempos de preparação associados à transição entre a última tarefa alocada na máquina ϕ^* (tarefa k) e as tarefas não alocadas, $s_{kj}^{\phi^*}$, bem como os tempos de preparação $s_{ij}^{\phi^*}$ e $s_{ji}^{\phi^*}$, em que i e $j \in \overline{N}$. Tempos de preparação entre tarefas já alocadas e/ou aqueles associados a outras máquinas que não ϕ^* não exercem influência no cálculo do índice de prioridade e por este motivo são desconsiderados no cálculo de $\overline{s}$.

Índice de prioridade ATCS

Para cada tarefa não alocada j, o índice de prioridade ATCS é calculado segundo a Equação 3.6. No índice ATCS, a primeira exponencial atribui prioridade máxima para as tarefas atrasadas $(d_j - p_j^{\phi} - t^* < 0)$, independentemente de quão atrasadas elas estejam. Para as demais, a prioridade atribuída é tanto menor quanto mais longe a data de entrega estiver do instante t^* de liberação da máquina ϕ^*. A segunda exponencial, por sua vez, prioriza tarefas que exijam menores tempos de preparação. Os fatores de *look-ahead*, k_1 e k_2, estão definidos

no Capítulo 2. A aplicação do índice ATCS, a problemas com penalizações de avanço e de atraso e nos quais as tarefas não estão todas liberadas no instante inicial do horizonte de planejamento, pode ser justificada observando-se que, a partir de um determinado ponto do procedimento de construção da solução inicial é bem provável que as tarefas não alocadas já estejam atrasadas e, conseqüentemente, já estejam liberadas para processamento.

$$I_j\left(t^*,k\right) = \frac{t_j}{p_j^{\phi}}\exp\left(-\frac{\max((d_j - p_j^{\phi} - t^*),\quad 0)}{k_1\overline{p}}\right)\exp\left(-\frac{s_{kj}^{\phi}}{k_2\overline{s}}\right) \tag{3.6}$$

Índice de prioridade ATCS+E

Com este índice, proposto no presente trabalho, procura-se levar em consideração as penalizações por avanço das tarefas não alocadas. Considere que a data de entrega d_j da tarefa não alocada j é tal que $(d_j - p_j^{\phi} - s_{kj}^{\phi} - t^*) \leq 0$. Neste caso, a tarefa j não fica adiantada ao ser alocada na última posição da máquina ϕ*. O índice de prioridade da referida tarefa é calculado segundo a Equação 3.6 (índice ATCS). Por outro lado, se $(d_j - p_j^{\phi} - s_{kj}^{\phi} - t^*) > 0$ a alocação da tarefa j fica sujeita a um custo de avanço. O índice de prioridade é, então, dado pela Equação 3.7 (índice ATCS+E). Como no índice ATCS, priorizam-se as tarefas com maiores custos de atraso, menores tempos de processamento e que requeiram menores tempos de preparação para serem executadas imediatamente após a última tarefa alocada. Além disso, no índice ATCS+E as prioridades de tarefas com maiores custos de avanço são diminuídas.

$$I_j\left(t^*,k\right) = \frac{t_j}{p_j^{\phi}}\exp\left(-\frac{(d_j - p_j^{\phi} - t^*)}{k_1\overline{p}}\right)\exp\left(-\frac{s_{kj}^{\phi}}{k_2\overline{s}}\right) \quad - \quad \frac{e_j}{p_j^{\phi}}\left(1 - \exp\left(-\frac{s_{kj}^{\phi}}{k_2\overline{s}}\right)\right) \tag{3.7}$$

Índice de prioridade MATCS

O índice MATCS (*Modified Apparent Tardiness Cost with Setups*) proposto por Kim e Shin (2003), para o problema de minimização do máximo *lateness*, leva em consideração os instantes de liberação das tarefas não alocadas. Ele é dado pela Equação 3.8, em que $s_{kj}^{\bullet}$ é o tempo real necessário entre a conclusão da última tarefa alocada (tarefa k) e o início efetivo da tarefa j. Se a tarefa j já estiver liberada no instante t*, faz-se $s_{kj}^{\bullet} = s_{kj}^{\phi}$. Por outro lado, o tempo de preparação é acrescido da diferença entre o instante de liberação da tarefa j e o instante t*, isto é, $s_{kj}^{\bullet} = s_{kj}^{\phi} + r_j - t^*$.

$$I_j\left(t^*, k\right) = \frac{1}{p_j^{\phi}} \exp\left(-\frac{\max((d_j - p_j^{\phi} - t^*), \quad 0)}{k_1 \overline{p}}\right) \exp\left(-\frac{s_{kj}^{\bullet}}{k_2 \overline{s}}\right) \tag{3.8}$$

Se o instante de liberação da tarefa j for menor que o instante de conclusão da tarefa k, a preparação para a tarefa j pode ser iniciada de imediato. Caso contrário, um intervalo de tempo dado por $r_j - t^*$ precisa ser aguardado. Tal intervalo de tempo é somado ao tempo de preparação, para efeito de cálculo do índice de prioridade da tarefa j. As tarefas ainda não liberadas têm o índice de prioridade reduzido, de forma tão mais intensa quanto maior a diferença $r_j - t^*$.

No problema considerado no presente trabalho a função objetivo envolve penalizações por atraso, t_j, específicas para cada tarefa. Considerar tal particularidade exige uma pequena modificação no índice MATCS, como proposto na Equação 3.9.

$$I_j\left(t^*, k\right) = \frac{t_j}{p_j^{\phi}} \exp\left(-\frac{\max((d_j - p_j^{\phi} - t^*), \quad 0)}{k_1 \overline{p}}\right) \exp\left(-\frac{s_{kj}^{\bullet}}{k_2 \overline{s}}\right) \tag{3.9}$$

Além disso, a definição original do termo $s_{kj}^{\bullet}$, apresentada por Kim e Shin (2003), é adequada para ambientes nos quais a preparação para o processamento de cada tarefa não pode ser iniciada antes de seu instante de liberação, r_j. Nos ambientes em que esta restrição não é imposta, uma definição mais apropriada para o termo $s_{kj}^{\bullet}$ pode ser estabelecida. Se a tarefa j já estiver liberada no instante $t^* + s_{kj}^{\phi}$, então, $s_{kj}^{\bullet} = s_{kj}^{\phi}$, pois a tarefa pode ser iniciada logo após a conclusão de sua preparação. Do contrário, um tempo ocioso dado por $r_j - \left(t^* + s_{kj}^{\phi}\right)$ precisa ser aguardado e neste caso, $s_{kj}^{\bullet} = r_j - t^*$.

Testes preliminares indicaram um desempenho melhor da função gulosa baseada no índice MATCS. A função gulosa é definida na Equação 3.10, em que $I_{\max}$ representa o máximo valor do índice de prioridade. A lista restrita de candidatas é composta pelas tarefas cujos valores da função satisfaçam a Equação 3.11, com o parâmetro de aleatorização, α, definido no intervalo $[0,1]$ e escolhido com probabilidade uniforme a cada iteração. A probabilidade de escolha de uma tarefa da lista restrita de candidatas (LRC), a cada passo do processo de construção da solução inicial é dada pela Equação 3.12.

$$g_j(t^*,k) = \frac{I_j(t^*,k)}{I_{\max}} \tag{3.10}$$

$$g_j(t^*,k) \in \left[g_{\min} + (1-\alpha)(g_{\max} - g_{\min}) \quad , \quad g_{\max}\right] \tag{3.11}$$

$$p_j(t^*,k) = \frac{g_j(t^*,k)}{\sum_{r \in LRC} g_r(t^*,k)} \tag{3.12}$$

Após a alocação de cada tarefa, qualquer que seja o índice de prioridade utilizado na definição da função gulosa, executa-se o procedimento de inserção de tempos ociosos na máquina modificada.

3.3.2 Busca local

Assim como feito para o problema de minimização de atrasos (Capítulo 2), foram definidas duas variantes, uma empregando movimentos de troca e de inserção, de modo alternado, e outra empregando movimentos *cross* e *Or-Opt*, também alternadamente.

3.3.3 Estratégias para melhoria do desempenho do GRASP básico

A memória de longo prazo é empregada de dois modos distintos – influenciando a fase construtiva e na pós-otimização, por meio de religação de caminhos.

Fase construtiva com memória

A lista restrita de tarefas candidatas, no GRASP com memória, é definida sob a influência de uma população de soluções de elite (soluções de alta qualidade e suficientemente distintas). Para o problema considerado neste capítulo o ambiente é de máquinas não-relacionadas. Desse modo, tanto as relações de precedência imediata entre tarefas quanto as atribuições tarefas-máquinas são importantes para a definição da função de intensidade, $H_k(j)$, como indicado pela Equação 3.13. Nesta equação, a variável $\delta_1(S,k,j)$ assume o valor 1 se a tarefa k preceder imediatamente a tarefa j na solução de elite S. Caso contrário ela assume o valor zero. A variável $\delta_2(S,\phi^*,j)$ assume o valor 1 se a tarefa j for processada na máquina ϕ^*, na solução de elite S. Caso contrário ela assume o valor zero.

Deve ser observado que, em uma instância de dimensões n x m, $\delta_1(S,k,j)=1$ ocorre com probabilidade $1/n$, enquanto que a probabilidade de $\delta_2(S,\phi^*,j)=1$ é $1/m$. Como, em geral, $n >> m$, é razoável atribuir um peso maior a $\delta_1(S,k,j)$, como feito na Equação 3.13. O custo de cada solução de elite é representado por $Z(S)$ e o da melhor solução por $Z(S^*)$.

$$H_k(j) = n \sum_{s \in Elite} \delta_1(S,k,j)\frac{Z(S^*)}{Z(S)} \quad + m \sum_{s \in Elite} \delta_2(S,\phi^*,j)\frac{Z(S^*)}{Z(S)} \tag{3.13}$$

Antes de ser combinada com a função gulosa, a função de intensidade é dividida por $H_{\max}$, o máximo valor de $H_k(j)$, dando origem à função *h(j)* (Equação 3.14).

$$h(j) = \frac{H_k(j)}{H_{\max}} \tag{3.14}$$

A função de avaliação, $E(t^*, j)$, para cada tarefa j não alocada, é construída a partir de uma combinação convexa entre a função gulosa (definida no item 3.3.1) e a função de intensidade, ambas normalizadas, como indicado na Equação 3.15. Pesos são dados a cada uma de tais funções, de modo a controlar o grau de diversificação / intensificação ao longo das iterações.

$$E(t^*, j) = (1-\lambda).g(t^*, j) + \lambda.h(j) \tag{3.15}$$

O fator de ponderação, λ, presente na Equação 3.15, assume valores no intervalo [0,1] e controla o grau de intensificação ou de diversificação. Seu valor é definido de modo adaptativo, como explicado no Capítulo 2, em termos da função de entropia normalizada.

A lista restrita de tarefas candidatas é formada pelas tarefas que satisfazem a condição imposta pela Equação 3.16, em que $\alpha \in [0,1]$ é definido, a cada iteração, com probabilidade uniforme.

$$E(t^*, j) \in \left[E_{\min} + (1-\alpha)(E_{\max} - E_{\min}) \quad , \quad E_{\max}\right] \tag{3.16}$$

A probabilidade de escolha de uma tarefa da lista restrita de candidatas (LRC), a cada passo do processo de construção da solução inicial é definida pela Equação 3.17.

$$p_j(t^*, k) = \frac{E(t^*, j)}{\sum_{i \in LRC} E_i(t^*, j)} \tag{3.17}$$

O conjunto de elite é construído nos mesmos moldes apresentados no Capítulo 2.

Princípio de otimalidade próxima e Religação de caminhos

O princípio de otimalidade próxima (POP) e a religação de caminhos são aplicados exatamente como indicado no Capítulo 2.

3.4 Busca Tabu

3.4.1 Heurística construtiva

A heurística construtiva utilizada para a Busca Tabu é uma adaptação daquela desenvolvida por Mazzini (1995) para o problema de minimização de custos de avanço e de atraso em uma única máquina e sem tempos de preparação. Três são as etapas envolvidas:

a) ordenamento das tarefas, segundo valores não-decrescentes das médias dos instantes de início preferencial;
b) construção de uma solução parcial factível, alocando as tarefas uma a uma, segundo o ordenamento anterior;
c) atualização dos tempos ociosos.

Instante de início preferencial

Para uma dada tarefa j define-se o instante de início preferencial, u_j, como sendo aquele no qual a tarefa fica sujeita ao menor custo. Uma vez que as máquinas não são idênticas, a tarefa pode ter tempos de processamento e de preparação distintos em máquinas distintas. Conseqüentemente, uma tarefa pode ter distintos instantes de início preferencial, dependendo da máquina na qual ela venha a ser processada. Dessa forma, é necessário definir um instante de início preferencial da tarefa j, u_j^{ϕ}, em cada máquina, ϕ, como indicado na Equação 3.18, para, em seguida, definir o instante de início preferencial médio da tarefa – Equação 3.19. Na primeira condição da Equação 3.18 está implícita a possibilidade de início da preparação, antes que a tarefa j tenha sido liberada para processamento. Portanto, a Equação 3.18 é coerente com a definição do problema. Deve ser observado que o instante de início preferencial médio é empregado apenas para ordenar as tarefas. Durante a etapa de alocação, os custos (exatos ou não) envolvidos com a alocação da tarefa candidata levam em conta os inícios preferenciais específicos da referida tarefa em cada máquina.

$$u_j^{\phi} = \begin{cases} r_j, & \text{se} \quad r_j + p_j^{\phi} > d_j \\ d_j - p_j^{\phi}, & \text{se} \quad r_j + p_j^{\phi} \leq d_j \end{cases} \tag{3.18}$$

$$\overline{u}_j = \frac{1}{m} \sum_{\phi=1}^{m} u_j^{\phi} \tag{3.19}$$

Alocação das tarefas

O procedimento de construção de uma solução factível envolve duas sub-etapas. Na primeira, a alocação de uma tarefa é avaliada em cada uma das máquinas. Na segunda, escolhe-se a máquina à qual esteja associado o menor acréscimo no custo da solução parcial. Caso o acréscimo de custo observado em duas ou mais máquinas seja o mesmo, escolhe-se aquela que apresenta a menor carga de trabalho, que corresponde à soma dos tempos de processamento e de preparação.

A alocação da primeira tarefa é direta. Uma vez que todas as máquinas estão vazias, é possível alocar a primeira tarefa, em qualquer uma das máquinas, no instante de início preferencial correspondente, sem risco de tal alocação gerar uma infactibilidade, representada por sobreposição de tarefas.

Já a alocação de cada nova tarefa, a partir da segunda, exige que se façam análises no sentido de detectar e desfazer possíveis sobreposições entre a nova tarefa e outras anteriormente alocadas.

No caso de uma única máquina, e também para máquinas idênticas, o ordenamento das tarefas segundo valores não-decrescentes dos instantes de início preferencial permite afirmar que a alocação da nova tarefa j não provoca sobreposição se o seu instante de conclusão, $C_j = u_j^{\phi} + p_j^{\phi}$, satisfizer a condição imposta pela Equação 3.20, ou seja, se a tarefa j for alocada após a última tarefa da seqüência parcial.

$$C_j \geq C_{[última]} + s^{\phi}_{j[última]} + p_j^{\phi} \tag{3.20}$$

Na Equação 3.20 e nas outras que a seguem é utilizada a seguinte convenção:

- j : tarefa cuja posição ainda não está definida na j-ésima chamada do procedimento de alocação;
- $[i]$: tarefa que ocupa a i-ésima posição na seqüência parcial da máquina ϕ;
- $[última]$: tarefa que ocupa a última posição na seqüência parcial da máquina ϕ;
- C_j : instante de conclusão da tarefa j, calculado considerando que seu processamento comece no respectivo instante de início preferencial;
- $C_{[i]}$: instante efetivo de conclusão da i-ésima tarefa da seqüência parcial.

No caso de máquinas paralelas não-relacionadas, ao ordenar as tarefas segundo valores médios dos instantes de início preferencial, criam-se outras duas possibilidades de alocação de uma nova tarefa sem que haja sobreposição.

A nova tarefa pode ser alocada antes da primeira da seqüência parcial anterior, se $C_j + s^{\phi}_{j[1]} \leq C_{[1]} - p^{\phi}_{[1]}$, ou pode ser alocada entre duas outras tarefas, [i] e [$i+1$], para as quais as condições $C_{[i]} + s^{\phi}_{[i]j} \leq u^{\phi}_j$ e $C_j + s^{\phi}_{j[i+1]} \leq C_{[i+1]} - p^{\phi}_{[i+1]}$ sejam satisfeitas.

Se nenhuma das três situações anteriormente apresentadas for possível, a alocação da tarefa j, no intervalo de tempo que começa em seu instante de início preferencial, implica em sobreposição com alguma tarefa já alocada na máquina ϕ. A sobreposição deve ser desfeita por meio de um dos movimentos descritos a seguir, nos quais se considera que a nova tarefa, ou tarefa candidata, se sobrepõe, primariamente, à tarefa [i] da seqüência parcial.

Proposição. Seja [i] a primeira tarefa da seqüência parcial tal que $C_{[i]} + s^{\phi}_{[i]j} > u^{\phi}_j$. Há uma sobreposição entre j e [i], pois o intervalo de tempo entre a conclusão da tarefa [i] e o início preferencial de j não é suficiente para a preparação $s^{\phi}_{[i]j}$. Como conseqüência da desigualdade triangular dos tempos de preparação, tem-se $C_{[q]} + s^{\phi}_{[q]j} > u^{\phi}_j$, para toda tarefa [$q$], posterior à tarefa [$i$].

Demonstração: Seja (i) $C_{[i]} + s^{\phi}_{[i]j} > u^{\phi}_j$. Contrariando o enunciado da proposição, suponha que (*ii*) $C_{[i+k]} + s^{\phi}_{[i+k]j} \leq u^{\phi}_j$, para $k > 0$. Por definição, a solução parcial, antes da alocação da tarefa j, é factível. Portanto, (*iii*) $C_{[i]} + s^{\phi}_{[i][i+1]} + ... + s^{\phi}_{[i+k-1][i+k]} \leq C_{[i+k]}$. Substituindo (*iii*) em (*ii*), tem-se (*iv*) $C_{[i]} + s^{\phi}_{[i][i+1]} + ... + s^{\phi}_{[i+k-1][i+k]} + s^{\phi}_{[i+k]j} \leq u^{\phi}_j$. Combinando ($i$) e (*iv*), tem-se (*v*) $s^{\phi}_{[i]j} > s^{\phi}_{[i][i+1]} + ... + s^{\phi}_{[i+k-1][i+k]} + s^{\phi}_{[i+k]j}$, o que é possível apenas se a desigualdade triangular não for respeitada.

Conseqüentemente, para toda tarefa [v] anterior à tarefa [i] tem-se $C_{[v]} + s^{\phi}_{[v]j} < u^{\phi}_j$, pois, do contrário, [$i$] não seria a primeira tarefa da seqüência parcial tal que $C_{[i]} + s^{\phi}_{[i]j} > u^{\phi}_j$. Em especial, $C_{[i-1]} + s^{\phi}_{[i-1]j} < u^{\phi}_j$, o que significa que a tarefa j não se sobrepõe à [i-*1*]. Este resultado é importante porque a heurística construtiva é definida a partir da premissa de que o intervalo de tempo entre a conclusão da tarefa [i-*1*] e o início preferencial de j é suficiente para o tempo de preparação $s^{\phi}_{[i-1]j}$, por maior que seja tal tempo de preparação.

Movimento 1

Neste movimento, mostrado na Figura 3.4, a tarefa [i] perde a sua posição para a tarefa candidata j, que é alocada, no seu início preferencial, entre as tarefas [i-1] e [i+1]. A tarefa [i] é deslocada para a direita, de uma quantidade igual a $C_j - C_{[i]} + s^{\phi}_{j[i]} + p^{\phi}_{[i]}$, de modo que ela possa ser executada imediatamente após a tarefa j.

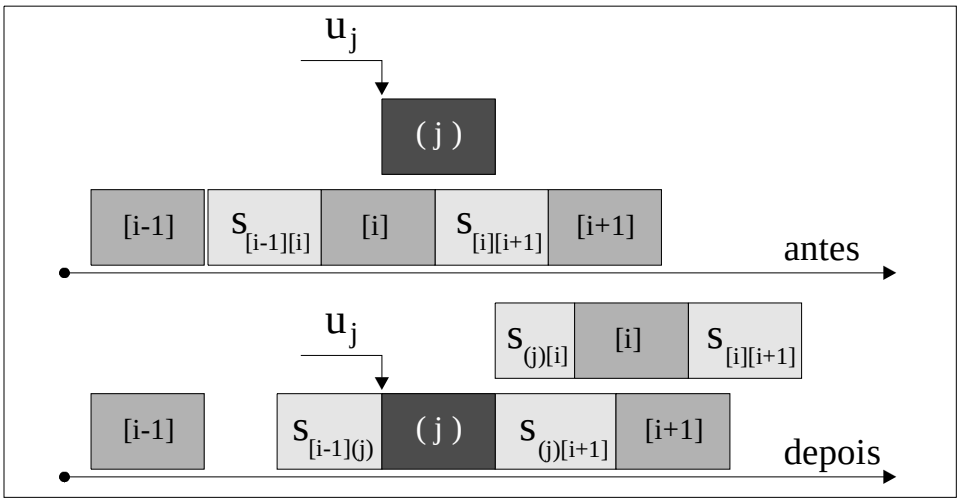

Figura 3.4 – PSET: Movimento 1 da heurística construtiva para a Busca Tabu

A variação aproximada do custo, decorrente da eventual execução do movimento 1, é dada pela Equação 3.21, em que $C'_{[i]} = C_j + s^{\phi}_{j[i]} + p^{\phi}_{[i]}$ é o novo instante de conclusão de [i].

$$\begin{aligned}\delta_1^{\phi} &= \max\left(t_{[i]}(C'_{[i]} - d_{[i]}), e_{[i]}(d_{[i]} - C'_{[i]})\right) \\ &\quad - \max\left(t_{[i]}(C_{[i]} - d_{[i]}), e_{[i]}(d_{[i]} - C_{[i]})\right) + \max\left(t_j(C_j - d_j), e_j(d_j - C_j)\right)\end{aligned} \tag{3.21}$$

Em sua nova condição de candidata, a tarefa [i] pode se sobrepor à tarefa [i+1]. Antes de resolver este novo conflito, e considerando que o movimento 1 tenha sido o escolhido, é preciso garantir que o intervalo entre j e [i+1] é suficiente para a execução da preparação entre estas duas tarefas. Neste sentido, se necessário, a tarefa [i+1] e todas as outras posteriores a ela são deslocadas à direita de modo que seja obtido o intervalo adequado. Esta situação é mostrada na Figura 3.4. Os custos decorrentes de tais deslocamentos, contudo, não são considerados no cálculo do custo aproximado do movimento 1.

Movimento 2

Neste movimento, indicado na Figura 3.5, a tarefa [i] é mantida em sua posição. A tarefa candidata, j, é deslocada para a direita, de uma quantidade igual a $C_{[i]} - C_j + s^{\phi}_{[i]j} + p^{\phi}_j$, para que possa ser executada imediatamente após a tarefa [i], passando, eventualmente, a disputar espaço com a tarefa [$i+1$].

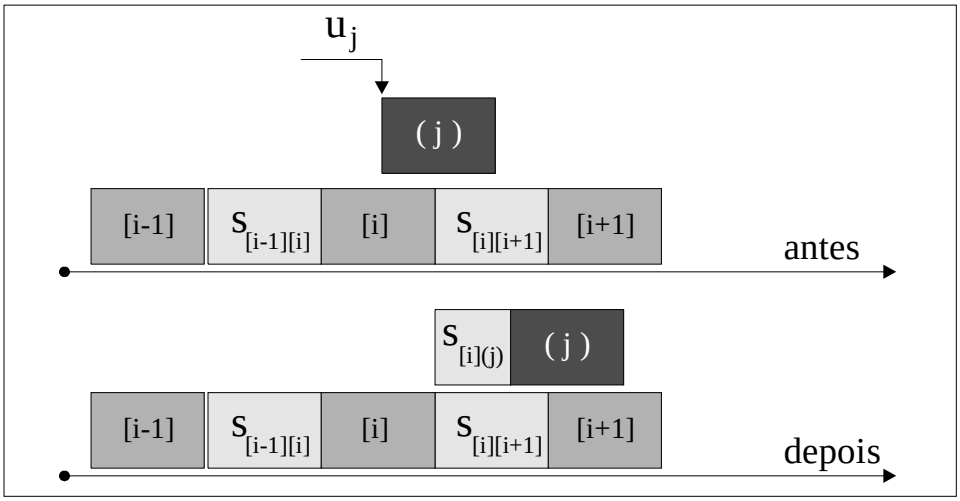

Figura 3.5 – PSET: Movimento 2 da heurística construtiva para a Busca Tabu

A variação do custo, decorrente da eventual realização do movimento 2, é dada pela Equação 3.22, em que $C'_j = C_{[i]} + S^{\phi}_{[i]j} + p^{\phi}_j$ é o novo instante de conclusão da tarefa j. A variação de custo é exata porque envolve apenas o deslocamento da tarefa j. Nenhuma tarefa já alocada é afetada.

$$\delta^{\phi}_2 = \max\left(t_j(C'_j - d_j), e_j(d_j - C'_j)\right) - \max\left(t_j(C_j - d_j), e_j(d_j - C_j)\right) \quad (3.22)$$

Movimento 3

Como no movimento 1, a tarefa [i] perde a sua posição para a tarefa candidata j. Contudo, força-se um deslocamento de j para a esquerda. O novo instante de início de j é limitado pela condição mais severa dentre aquelas mostradas a seguir, de modo a não impor uma penalização de avanço desnecessária à tarefa j. Deve ser observado que a condição **A** garante que o deslocamento à esquerda da tarefa j não provoca sobreposição com a tarefa [i-1]. A condição **B** garante o respeito ao instante de liberação, enquanto que a condição **C** avalia o quanto a tarefa j precisa ser antecipada para que sua alocação possa ocorrer em precedência imediata à tarefa [i].

A. $C_{[i-1]} + s^{\phi}_{[i-1]j}$

B. r_j

C. $C_{[i]} - p^{\phi}_{[i]} - p^{\phi}_{j} - s^{\phi}_{j[i]}$

Se o movimento 3, indicado na Figura 3.6, for realizado, os novos instantes de conclusão das tarefas j e [i] passam a ser aqueles indicados na Equação 3.23.

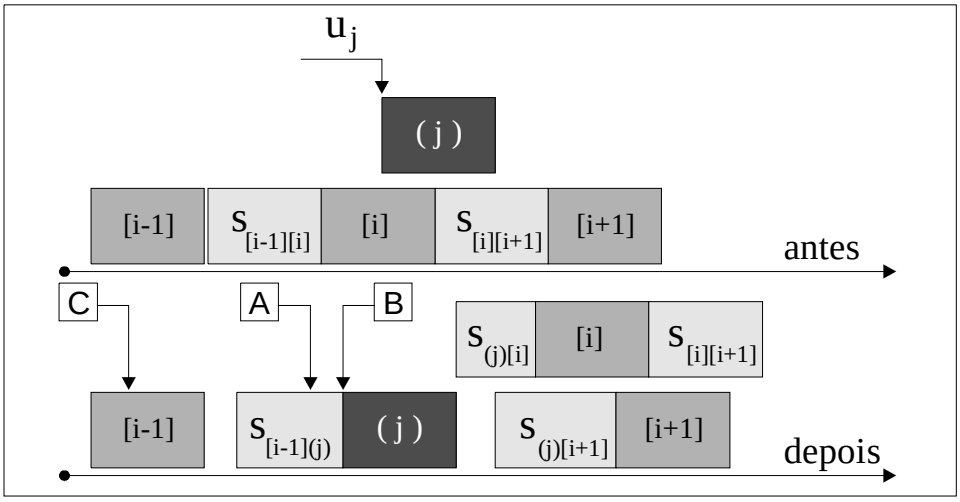

Figura 3.6 – PSET: Movimento 3 da heurística construtiva para a Busca Tabu

$$C'_{(j)} = \max(A, B, C) + p^{\phi}_{(j)} \quad \text{e} \quad C'_{[i]} = C'_{(j)} + s^{\phi}_{(j)[i]} + p^{\phi}_{[i]} \tag{3.23}$$

Se a condição mais severa não for a **C**, a tarefa [i] sofre um deslocamento para a direita, fato que pode gerar uma sobreposição entre [i] e [$i+1$]. Esta é a situação mostrada na Figura 3.6. A variação aproximada de custo, decorrente da eventual realização do movimento 3, é dada pela Equação 3.24.

$$\begin{aligned}\delta^{\phi}_{3} = {} & \max\left(t_{(j)}(C'_{(j)} - d_{(j)}), e_{(j)}(d_{(j)} - C'_{(j)})\right) - \max\left(t_{(j)}(C_{(j)} - d_{(j)}), e_{(j)}(d_{(j)} - C_{(j)})\right) \\ & + \max\left(t_{[i]}(C'_{[i]} - d_{[i]}), e_{[i]}(d_{[i]} - C'_{[i]})\right) - \max\left(t_{[i]}(C_{[i]} - d_{[i]}), e_{[i]}(d_{[i]} - C_{[i]})\right)\end{aligned} \tag{3.24}$$

Antes de resolver o eventual conflito entre a tarefa [i] e a tarefa [$i+1$], considerando que o movimento 3 tenha sido o escolhido, pode ser necessário deslocar a tarefa [$i+1$] para a direita, bem como todas as outras posteriores a ela, de modo a obter o adequado intervalo para a execução da preparação entre as tarefas j e [$i+1$]. Contudo, os custos decorrentes de tais deslocamentos não são considerados no cálculo do custo aproximado do movimento 3.

Movimento 4

No movimento 4 força-se um deslocamento da tarefa [i] para a esquerda. O novo instante de início de [i] é limitado pela condição mais severa dentre as mostradas a seguir.

- **A.** $C_{[i-1]} + s^{\phi}_{[i-1][i]}$
- **B.** $r_{[i]}$
- **C.** $C_{(j)} - p^{\phi}_{(j)} - p^{\phi}_{[i]} - s^{\phi}_{[i](j)}$

Enquanto a condição **A** garante que o deslocamento à esquerda da tarefa [i] não provoca sobreposição com a tarefa [$i-1$], a condição **B** garante o respeito ao instante de liberação e a condição **C** avalia o quanto a tarefa [i] precisa ser antecipada para que sua alocação possa ocorrer em precedência imediata à tarefa j.

Se o movimento 4 for realizado, os novos instantes de término das tarefas [i] e j passam a ser aqueles dados pela Equação 3.25. Se a condição mais severa não for a **C**, a tarefa j sofre um deslocamento para a direita, podendo haver sobreposição entre j e [$i+1$]. Esta situação é mostrada na Figura 3.7. Na Equação 3.24 tem-se a variação de custo decorrente da eventual realização do movimento 4. A variação de custo é exata porque não há necessidade de deslocar a tarefa [$i+1$] e posteriores.

$$C'_{[i]} = \max(A, B, C) + p^{\phi}_{[i]} \quad \text{e} \quad C'_{(j)} = C'_{[i]} + s^{\phi}_{[i](j)} + p^{\phi}_{(j)} \tag{3.25}$$

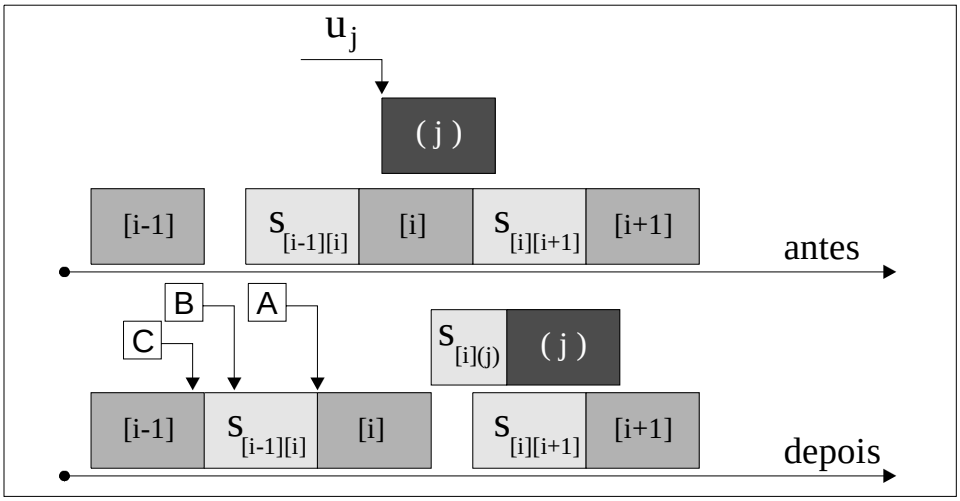

Figura 3.7 – PSET: Movimento 4 da heurística construtiva para a Busca Tabu

Uma vez avaliados cada um dos quatro movimentos, escolhe-se aquele que proporciona a menor variação de custo, δ^{ϕ}_{*}. O movimento escolhido é realizado sobre uma cópia temporária da solução parcial. Ele pode dar origem a uma nova sobreposição, desta vez envolvendo as tarefas j e [$i+1$] (movimentos 2 e 4) ou [i] e [$i+1$] (movimentos 1 e 3). Se isto ocorrer, o procedimento de análise de movimentos é executado novamente, até que toda sobreposição seja desfeita, antes de passar à alocação de mais uma nova tarefa.

Seja Π^{ϕ} a seqüência parcial da máquina ϕ, antes de se considerar a inserção da nova tarefa j e seja Π^{ϕ}_{j} a seqüência parcial acrescida de j. A variação aproximada total do custo, associada à inserção de j na seqüência parcial da máquina ϕ, é dada pela Equação 3.26, em que $\tilde{Z}(\Pi^{\phi}_{j})$ é o custo aproximado da seqüência Π^{ϕ}_{j} (calculado antes da atualização dos

tempos ociosos). A máquina escolhida para o processamento de j é aquela que fornece a menor variação aproximada de custo total. Tal máquina, denominada ϕ^*, tem a sua seqüência parcial atualizada com o conteúdo da cópia temporária da solução parcial, isto é, $\Pi^{\phi^*} \leftarrow \Pi_j^{\phi^*}$ e em seguida passa pelo procedimento de atualização de tempos ociosos (descrito no item 3.2).

$$\Delta Z_j^{\phi} = \tilde{Z}(\Pi_j^{\phi}) - Z(\Pi^{\phi}) \tag{3.26}$$

3.4.2 Memória de curto prazo

Um aspecto estrutural relevante para o problema tratado neste capítulo é o fato das máquinas serem não-relacionadas. Desse modo, a atribuição de uma tarefa a diferentes máquinas apresenta significados e impactos também diferentes (traduzidos por diferentes tempos de processamento e de preparação). Logo, para movimentos que alterem as atribuições das tarefas, é possível estabelecer um critério de proibição de movimentos baseado na atribuição – critério C_{atrib}. Para os movimentos internos, os critérios C_{in} e C_{out} (apresentados no Capítulo 2) permanecem inalterados.

Critério de proibição C_{atrib}

Seja M_3 uma matriz n x m cujos elementos, $M_3(i,\phi)$, armazenam a iteração até a qual o retorno da tarefa i para a máquina ϕ é proibido. De acordo com o critério de proibição C_{atrib}, o *status* do movimento é "Proibido" se $M_3(i,\phi)$ for maior do que a iteração (x) atual, para qualquer par (i,ϕ) tal que (i,ϕ) corresponda a uma atribuição a ser estabelecida. Como exemplo, considere que na iteração (x) atual, as tarefas i e j estejam atribuídas às máquinas ϕ_1 e ϕ_2, respectivamente. Um movimento de troca entre estas duas tarefas tem o *status* "Proibido" se: $\left(M_3(i,\phi_2) > x\right)$ ou $\left(M_3(j,\phi_1) > x\right)$.

Duração tabu e critério de aspiração

A estratégia de definição da duração tabu e o critério de aspiração são os mesmos adotados para o problema tratado no Capítulo 2.

3.4.3 Memória de longo prazo

As estratégias de longo prazo são, em essência, as mesmas adotadas para o problema tratado no Capítulo 2. A única alteração a ser realizada é na variante derivada do procedimento de Rochat e Taillard (1995), por conta do fato das máquinas serem não-relacionadas. Na construção do conjunto T de seqüências é necessário referenciar a máquina específica que processa cada seqüência. Após escolher uma seqüência do conjunto T, devem-se eliminar todas as seqüências que processem ao menos uma tarefa da seqüência escolhida, bem como todas as seqüências que estejam atribuídas à mesma máquina da seqüência escolhida (passo 7e). Além disso, na factibilização (passo 9b) emprega-se o índice de prioridade MATCS.

3.4.4 Pós-otimização

Como estratégia de pós-otimização, emprega-se religação de caminhos (regressiva e progressiva) entre cada uma das soluções do conjunto de elite e a solução incumbente, atualizando-se o conjunto de elite, sempre que possível. O processo é aplicado enquanto houver atualizações no conjunto de elite.

Capítulo 4

Testes computacionais: Problema PST

4.1 Instâncias e critério de parada

As instâncias são geradas segundo um procedimento que tem por base aquele adotado em Lee e Pinedo (1997). Ao todo são 360 instâncias com tamanhos variados. Há 24 instâncias para cada combinação de máquinas, $m = (2,3,4,6,12)$, e tarefas, $n = (60,90,120)$. Todos os parâmetros característicos são escolhidos com probabilidade uniforme. Os tempos de processamento estão distribuídos no intervalo $[50,150]$, com média 100. As penalizações por unidade de tempo de atraso, no intervalo $[1,100]$. Na geração das datas de entrega e dos tempos de preparação são utilizados os fatores:

- Faixa (espalhamento) das datas de entrega, $R = (d_{\max} - d_{\min}) / C_{\max}$;
- Fator de aperto das datas de entrega, $\tau = 1 - (\overline{d} / C_{\max})$;
- Fator de severidade dos tempos de preparação, $\eta = \overline{s} / \overline{p}$;

Nestes fatores, $d_{\min}$, $\overline{d}$ e $d_{\max}$ representam, respectivamente os valores mínimo, médio e máximo das data de entrega, $\overline{s}$ é a média dos tempos de preparação, $\overline{p}$ é a média dos tempos de processamento e $C_{\max}$ é uma estimativa do *makespan* (Equação 4.1).

$$C_{\max} = \frac{n}{m}\left[\overline{p} + \overline{s}\left(0{,}4 + \frac{10m^2}{n^2} - \frac{\eta}{7}\right)\right] \qquad (4.1)$$

As instâncias geradas dividem-se em apertadas (τ = 0.9) e folgadas (τ = 0.3). O fator de espalhamento das datas de entrega, R, assume os valores 0.25 (datas de entrega mais concentradas) e 0.75 (datas de entrega mais espalhadas). Para o fator de severidade dos tempos de preparação são adotados os valores (η = 0.25) e (η = 0.75). As datas de entrega estão distribuídas uniformemente no intervalo $\left[(1-R)\bar{d},\ \bar{d}\right]$ com probabilidade τ e no intervalo $\left[\bar{d},\ \bar{d}+(C_{\max}-\bar{d})R\right]$ com probabilidade (1 - τ). Os tempos de preparação ficam contidos no intervalo $\left[\frac{2}{3}\eta\bar{p},\ \frac{4}{3}\eta\bar{p}\right]$, garantindo-se, dessa forma, a desigualdade triangular. Todas as tarefas estão liberadas no instante zero. Para cada combinação dos fatores τ, R e η foram geradas 3 instâncias. Na Tabela 4.1 mostra-se um resumo das dimensões e dos parâmetros característicos do conjunto de instâncias.

Tabela 4.1 – Dimensões das instâncias e parâmetros característicos	
Tarefas	60, 90, 120
Máquinas	2, 3, 4, 6, 12
Fator τ (aperto das datas de entrega)	0.3 e 0.9
Fator R (espalhamento das datas de entrega)	0.25 e 0.75
Fator η (severidade dos tempos de preparação)	0.25 e 0.75

O critério de parada está definido em termos de um número máximo de soluções investigadas, que depende da soma dos tamanhos das vizinhanças de troca e de inserção. Este critério é adotado, também, para as variantes em que são empregados movimentos *cross* e *Or-Opt*. O tamanho da vizinhança de troca é dado por $V_{troca} = n(n-1)/2$, enquanto que a de inserção é $V_{inserção} = n(n+m-2)-(n-m)$. Para o cálculo da vizinhança de inserção, deve ser observado que cada uma das n tarefas pode ser inserida em $(n+m-2)$ posições e $(n-m)$ duplicações são causadas pela inserção das tarefas em posições adjacentes. Na Tabela 4.2 são apresentados os totais de soluções investigadas para cada tamanho de instância considerado. Os valores apresentados correspondem a 4000 vizinhanças completas de troca e de inserção.

Em comparação com a dimensão do espaço de solução (**Apêndice I**), o total de soluções investigadas é cerca de 10^{77} vezes menor (instâncias de 60 tarefas) e 10^{200} vezes menor (instâncias de 120 tarefas).

Tabela 4.2 – Totais de soluções investigadas			
	Tarefas		
Máquinas	60	90	120
2	21.248.000	48.068.000	85.688.000
3	21.492.000	48.432.000	86.172.000
4	21.736.000	48.796.000	86.656.000
6	22.224.000	49.524.000	87.624.000
12	23.688.000	51.708.000	90.528.000

Os códigos foram implementados em linguagem C++ e foram compilados no Microsoft Visual C++ 6.0. Todos os testes foram realizados em um computador com processador AMD Athlon de 1.0 GHz, com 512MB de memória RAM.

4.2 Medida de desempenho - Teste de Wilcoxon

Sejam X_A e X_B as séries de resultados obtidos pelas heurísticas A e B, quando aplicadas a um mesmo conjunto de q instâncias de um dado problema. No caso presente, X_A e X_B representam desvios percentuais medidos em relação ao melhor resultado conhecido para cada uma das q instâncias. Seja $D = X_A - X_B$. A hipótese nula do teste estatístico de Wilcoxon (Golden e Stewart, 1985) é H_0: $\mu_D = 0$, em que μ_D é a média dos valores em D. A hipótese nula caracteriza-se por considerar que as duas heurísticas, A e B, têm desempenhos semelhantes, não havendo dominância de uma sobre a outra.

No primeiro passo do teste os valores nulos em D são eliminados. Seja p o total de valores nulos. Os valores restantes (q-p) são classificados pelos seus valores absolutos. O elemento de menor valor absoluto recebe a classificação 1, enquanto o de maior valor a classificação (q-p). Ocorrendo empates em termos dos valores absolutos, calcula-se a média das classificações dos itens envolvidos no empate e cada um deles recebe a classificação média calculada. Em seguida, cada classificação não nula recebe um sinal positivo se $X_A - X_B > 0$ ou negativo, caso contrário. Finalmente, calcula-se a soma (W) de todas as classificações, com os respectivos sinais. A hipótese nula, segundo a qual as heurísticas têm desempenhos semelhantes, será rejeitada se $|W| > W_{crítico}$, em que $W_{crítico}$ pode ser calculado de

forma aproximada pela Equação 4.2, quando o número de elementos $(q-p) \geq 10$. Em se tratando de problemas de minimização, e sendo $D = X_A - X_B$, a rejeição da hipótese nula significa, dentro da margem de confiança estabelecida no cálculo de $W_{crítico}$, que a heurística A é pior que a B, caso $W > W_{crítico}$. Por outro lado, se W < 0 e $|W| > W_{crítico}$, tem-se que a heurística B é pior do que a A. Para um grau de confiança, α, de 95%, $Z(\alpha) = 1.645$ é o ponto da curva de distribuição normal tal que 95% da área sob a curva fica à esquerda do referido ponto.

$$W_{crítico} = Z(\alpha)\sqrt{(q-p)(q-p+1)(2q-2p+1)/6} \tag{4.2}$$

Na Tabela 4.3 apresenta-se um exemplo de aplicação do teste de Wilcoxon a um conjunto de 12 instâncias quaisquer. Em tal exemplo, o valor absoluto da soma das classificações com sinal é menor que o valor crítico para $(q-p) = 12$. Portanto, não há dominância de uma heurística sobre a outra.

Tabela 4.3 – Exemplo de aplicação do teste de Wilcoxon

Min	A	B	$\frac{100*(A-B)}{Min}$	Classificação absoluta	Classificação com sinal
1789779	1791290	1790834	0,025	3	3
2237852	2239965	2240476	-0,023	2	-2
1443882	1445989	1445507	0,033	4	4
2491519	2496247	2496303	-0,002	1	-1
2200747	2201896	2203007	-0,050	8	-8
2212705	2221327	2217943	0,153	11	11
1869121	1869528	1871701	-0,116	10	-10
2084018	2085167	2087001	-0,088	9	-9
1827834	1829617	1830367	-0,041	7	-7
2659633	2663562	2662572	0,037	5	5
2253075	2256244	2255397	0,038	6	6
2005382	2010241	2006270	0,198	12	12
				Soma	4
				$W_{Crítico}$	41,9

4.3 Caracterização das variantes e testes preliminares

Os testes preliminares tiveram por objetivo a calibração dos parâmetros de controle presentes em cada variante proposta para as meta-heurísticas GRASP e Busca Tabu. Também serviram para balizar a escolha dos tipos de movimentos (trocas e inserções) *versus* (*cross* e *Or-Opt*). Foram conduzidos sobre um conjunto reduzido de instâncias (aquelas de 60 tarefas, com 3 e 6 máquinas).

GRASP Básico (GB)

- Função gulosa: Baseada no índice ATCS (Equação 2.1);
- Parâmetro α (Equação 2.5): Escolhido aleatoriamente com probabilidade uniforme nos intervalos $[0.0, 1.0]$, $[0.1, 0.5]$ e $[0.5, 0.9]$. Os melhores resultados, medidos em termos de desvios percentuais em relação às melhores soluções conhecidas e em número de vitórias, foram obtidos com $\alpha \in [0.1, 0.5]$;
- Tarefas da LRC: Escolhidas com probabilidade diretamente proporcional ao valor da função gulosa (Equação 2.6);
- Busca local: Foi constatada uma superioridade da variante com movimentos *cross* e *Or-Opt* sobre a variante com movimentos de troca e inserção;
- Parâmetro ξ de redução de vizinhança: Foram avaliados os valores de $\xi = 0.0$ (vizinhança completa), 0.3, 0.6 e 0.9. Em termos da qualidade dos resultados alcançados não houve diferença significativa entre eles. No entanto, o custo computacional de cada iteração é menor quando $\xi = 0.9$. Portanto, este último foi o valor escolhido.

GRASP com Memória (GM)

- Função gulosa, parâmetro α, parâmetro ξ de redução de vizinhança (os mesmos utilizados em **GB**);
- Busca local: Vizinhanças alternadas *cross* e *Or-Opt*;
- Duração da fase de aprendizado: 20% do total de soluções a serem investigadas;
- Função de intensidade: Calculada segundo a Equação 2.7;
- Tarefas da LRC: Escolhidas com probabilidade diretamente proporcional ao valor da função gulosa (Equação 2.11);

- Parâmetro λ para ponderação entre a função gulosa e a função de intensidade: Definido a partir da função de entropia (Equação 2.12), com valores no intervalo 0.1 a 0.9 e incrementos de 0.1. A função de entropia é reavaliada a cada $x_1 = 10$ iterações (reinício + busca local);
- Cardinalidade do conjunto de elite, $|\Gamma|$: Foram avaliados conjuntos com 10, 20 e 30 soluções de elite. Os melhores resultados foram obtidos com $|\Gamma| = 10$;
- Limiar de distância, $D_{\min}$, entre soluções de elite: Foram avaliados os valores 0.3, 0.5 e 0.7, não tendo sido observados desempenhos muito diferentes entre eles. Optou-se, dessa forma, pelo valor intermediário, $D_{\min} = 0.5$;
- Parâmetro θ, que limita o fator de escala do critério de aceitação de soluções no conjunto de elite: Foram avaliados os valores $\theta = 1.0$ e $\theta = 0.5$. Os resultados alcançados com $\theta = 0.5$ foram ligeiramente melhores.

GM + Religações de caminhos na pós-otimização (GMP)

- Trajetórias de religação: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: *cross* (corresponde à variante RC2 descrita no Capítulo 2, item 2.3.3);
- Buscas locais ao longo das trajetórias: Vizinhanças *cross* e *Or-Opt*, exploradas alternadamente;
- Demais parâmetros assumem os valores estabelecidos para **GM**.

GM + POP na fase de aprendizado (GMPop)

- Ponto de aplicação da busca local sobre a solução parcial: Ao alocar 75% das tarefas. A aplicação em outros percentuais (menores) produz um aumento do esforço computacional, sem melhoria compatível nos resultados finais;
- Buscas locais sobre a solução parcial: Vizinhanças *cross* e *Or-Opt*;
- POP aplicado apenas nas iterações da fase de aprendizado. A aplicação em todas as iterações é muito cara. Além disso, experimentos anteriores (Armentano e França Filho, 2007) mostraram que a aplicação do POP nas iterações com memória pode requerer que um menor peso (λ) seja dado à função de intensidade, sob pena de comprometer a diversidade do procedimento;
- Demais parâmetros assumem os valores estabelecidos para **GM**.

GMPop + Religações de caminhos na pós-otimização (GMPP)

- Parâmetros assumem valores definidos para as variantes **GM**, **GMP** e **GMPop**.

Busca Tabu de Curto Prazo (TC)

- Solução inicial: Baseada no índice ATCS, com α = 0 (estritamente gulosa);
- Buscas locais e transição entre vizinhanças: Foi observado um melhor desempenho da variante baseada em movimentos *cross* e *Or-Opt* (executados alternadamente). A vizinhança *cross* tem tamanho que é proporcional a $n * m$, enquanto que a vizinhança *Or-Opt* cresce com $n \div m$. Desse modo, foi definido que a transição dos movimentos *cross* para os movimentos *Or-Opt* (uma vez detectada a estagnação local da busca) deveria ocorrer após um submúltiplo do produto $n * m$, enquanto que a transição em sentido contrário deveria ocorrer após um submúltiplo da razão $n \div m$. Os testes realizados indicaram os melhores resultados associados à transição *cross* / *Or-Opt* após $(n * m) \div 60$ iterações sem melhoria na solução de partida e associados à transição *Or-Opt* / *cross* após $n \div m \div 5$ iterações com estagnação (em cada máquina);
- Parâmetro ξ de redução de vizinhança: Foi utilizado o valor 0.9 (mesmo do GRASP);
- Critério de proibição e duração tabu: Extensivos testes foram realizados para definir a melhor combinação entre critério de proibição e duração tabu. Os melhores resultados foram alcançados com o critério de proibição C_{in}, que proíbe movimentos que promovem o retorno de arcos recentemente retirados da solução. Para os movimentos *cross*, o intervalo de duração tabu mais adequado foi $[0.0, 0.2 * n * m]$, enquanto que para os movimentos *Or-Opt* o intervalo mais adequado foi $[0.0, n \div m]$;
- Critério de aspiração: Um movimento tabu é admitido se permitir melhorar a solução incumbente.

TC + Reinícios segundo Rochat&Taillard (TRT)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo: Ocorre após um certo número de alternâncias entre movimentos *cross* e movimentos *Or-Opt* (ciclos de curto prazo), sem melhoria da solução incumbente. Os testes preliminares mostraram que instâncias mais folgadas, tanto em termos do fator de aperto das datas de entrega (τ), quanto em termos da relação tarefas/máquinas (μ) se beneficiam da ativação mais freqüente da estratégia de longo prazo. Desse modo, a ativação da estratégia de longo prazo foi definida para ocorrer após $c.\tau.\mu$ ciclos de curto prazo com estagnação da incumbente. Nos testes realizados, a constante c assumiu os valores 0.3, 1.5 e 3.0. Os melhores resultados foram encontrados para $c = 1.5$;
- A estratégia de Rochat e Taillard opera sobre um banco ($\boldsymbol{T}$) de seqüências de tarefas alocadas a uma mesma máquina, construído a partir de um conjunto Γ de soluções de elite, idêntico àquele adotado no GRASP com memória. A cardinalidade do conjunto de elite foi fixada em 10 e o limiar de distância entre soluções ($D_{\min}$) foi fixado em 0.5 (mesmos valores utilizados no GRASP com memória).

TRT + Religações de caminhos na pós-otimização (TRTP)

- Trajetórias de religação na pós-otimização: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: *cross;*
- Buscas locais ao longo das trajetórias: Vizinhanças *cross* e *Or-Opt*;
- Demais parâmetros assumem os valores estabelecidos para a variante **TRT**.

TC + Diversificação por freqüência de residência (TDiv)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo: Segue critério adotado para a variante **TRT**;
- Duração da fase de diversificação e intensidade da penalização (fator γ): A duração da fase de diversificação é definida em termos de números de ciclos de alternância entre movimentos *cross* e movimentos *Or-Opt*. Foram realizados diversos testes, com durações de 2, 4, 6 e 8 ciclos. Para cada duração da fase de diversificação o fator γ assumiu os valores de 0.1 a 1.0, com incrementos de 0.1. Os melhores resultados foram alcançados com a duração de 2 ciclos e $\gamma = 0.1$.

TDiv + Religações de caminhos intermediárias (TDivRC)

- Fases de curto prazo e de diversificação com os mesmos parâmetros empregados na variante **TDiv**;
- Solução de elite de referência para a realização da religação de caminho: a mais distante em relação à solução de mínimo local obtida após o término da fase de diversificação. O objetivo é explorar caminhos mais longos;
- Trajetórias de religação : Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias : *cross*;
- Buscas locais ao longo das trajetórias : Vizinhanças *cross* e *Or-Opt*.

TDivRC + Religações de caminhos na pós-otimização (TDivRCP)

- Trajetórias de religação na pós-otimização : Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias : *cross*;
- Buscas locais ao longo das trajetórias : Vizinhanças *cross* e *Or-Opt*;
- Demais parâmetros assumem valores estabelecidos para a variante **TDivRC**.

TC + Religações de caminhos intermediárias (TRC)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo : Segue o mesmo critério adotado nas variantes **TRT** e **TDiv**;
- Solução de elite de referência para a realização da religação de caminho: a mais distante em relação à solução atual;
- Trajetórias de religação : Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias : *cross*;
- Buscas locais ao longo das trajetórias : Vizinhanças *cross* e *Or-Opt*.

O resultado mais relevante da fase de testes preliminares foi o desempenho superior alcançado com a utilização dos movimentos *cross* e *Or-Opt*, em comparação àquele obtido com os movimentos de troca e de inserção, tanto no GRASP quanto na Busca Tabu. Resultado qualitativamente semelhante já havia sido obtido por Armentano e França Filho (2007), para o caso específico do GRASP, em um problema de minimização de soma de custos de atraso em um ambiente de máquinas paralelas idênticas.

4.4 GRASP (GB) *versus* GRASP (GM)

O GRASP com Memória (GM) vence o GRASP Básico (GB) em 359 das 360 instâncias testadas e perde em apenas uma (de 60 tarefas e 12 máquinas). A aplicação do teste de Wilcoxon revela (de modo óbvio) que GM domina GB, quaisquer que sejam os subconjuntos de instâncias considerados (instâncias de mesmas dimensões, instâncias com mesmo número de tarefas e mesmo parâmetro τ, etc.).

Na Figura 4.1(a) apresentam-se os desvios percentuais, calculados como $\left[Z(GB) - Z(GM)\right] / Z(\min)$, para as instâncias de 60 tarefas. No cálculo do desvio, $Z(GB)$ representa o custo obtido pelo GRASP básico, $Z(GM)$ representa o custo obtido pelo GRASP com memória e $Z(\mathbf{min})$ representa o menor custo conhecido para cada instância. Desvios positivos indicam, portanto, vitórias de GM sobre GB. As instâncias estão agrupadas em função do parâmetro τ e ordenadas em função de valores crescentes dos desvios percentuais. Como pode ser observado, os desvios percentuais são mais expressivos nas instâncias com maior média das datas de entrega, isto é, as mais folgadas (τ = 0.3). Para estas, o menor desvio foi de 0.23%. A média dos desvios foi de 2.75%, enquanto que o desvio máximo foi de 10.81%. Estas informações são apresentadas na Figura 4.1(a) como "τ = 0.3 : (0.23, 2.75, 10.81) %". Em relação ao conjunto de instâncias com τ = 0.9, observa-se uma instância na qual GM perde para GB (desvio negativo de -0.0039%). Na média, a superioridade de GM se traduziu em um desvio de 0.21%, com o maior desvio alcançando 0.57%. Comportamento semelhante é observado também para as instâncias de 90 tarefas, Figura 4.1(b), e 120 tarefas, Figura 4.1(c). Cabe observar que para instâncias de mesmas dimensões, aquelas com τ = 0.3 apresentam, em geral, custos menores, o que colabora para que os desvios percentuais sejam maiores.

Na análise focada no espalhamento das datas de entrega, Figura 4.2, observam-se desvios um pouco mais acentuados nas instâncias em que as datas de entrega são mais espalhadas em torno da data de entrega média, isto é, R = 0.75. Por exemplo, para as instâncias de 60 tarefas e R = 0.75 o desvio mínimo foi de –0.0039%, enquanto que a média dos desvios foi de 1.80% e o desvio máximo foi de 10.81%. Já para as instâncias com datas de entrega mais concentradas, R = 0.25, o menor desvio foi de 0.06%, a média foi de 1.16% e o desvio máximo chegou a 4.46%. Este comportamento é observado também para as instâncias de 90 e 120 tarefas.

Na Figura 4.3 são apresentados os desvios quando as instâncias são agrupadas em termos do fator de severidade dos tempos de preparação (fator η). Observam-se maiores médias de desvios, favoráveis a GM, associadas às instâncias com η = 0.75. Para as instâncias de 60 tarefas, o fator η é o que menor influência exerce no desempenho de GM em comparação ao de GB (curvas são bastante coincidentes).

O valor médio dos tempos de CPU gastos pelo GRASP Básico para as instâncias de 60 tarefas foi de 72s, enquanto que o GRASP com Memória, gastou, em média, 78s. Um acréscimo médio de 8.96%, para avaliar o mesmo número de soluções. Para as instâncias de 90 tarefas, GB gastou, em média, 281s, contra 308s de GM (aumento de 9.34%). Para as instâncias de 120 tarefas, o tempo médio gasto por GB foi de 715s, enquanto que GM gastou 775s (aumento de 8.45%).

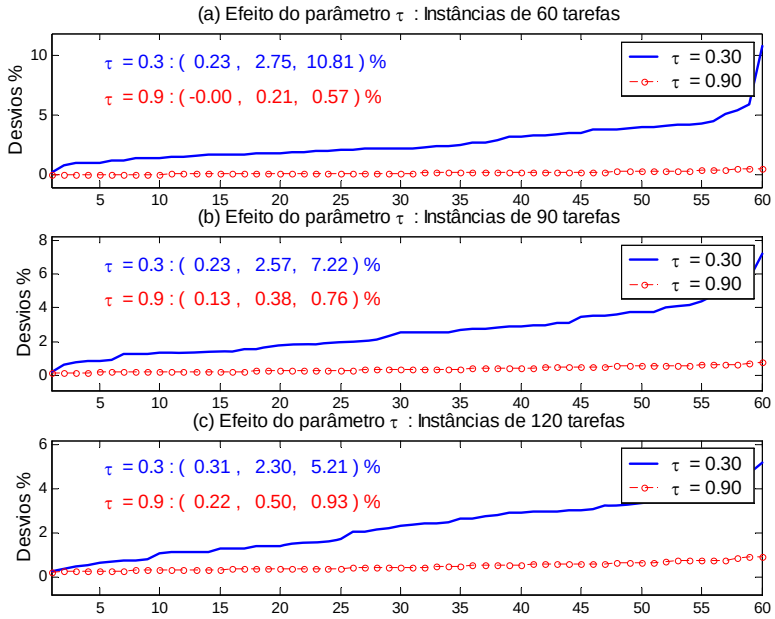

Figura 4.1 – PST: GB *versus* GM – Efeito do parâmetro τ

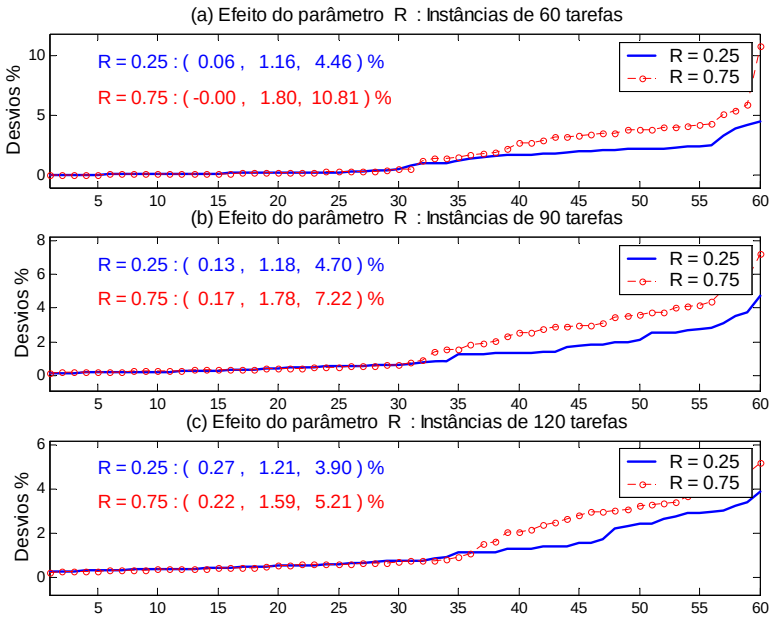

Figura 4.2 – PST: GB *versus* GM – Efeito do parâmetro R

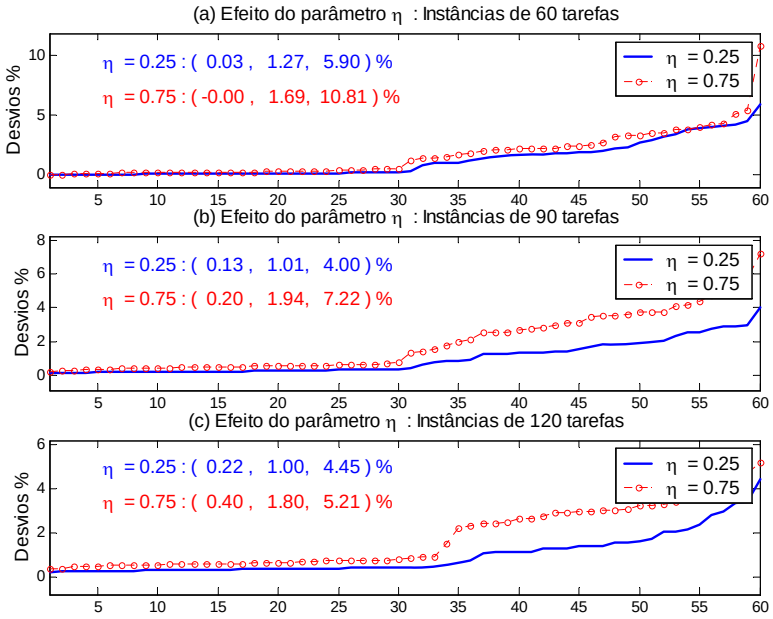

Figura 4.3 – PST: GB *versus* GM – Efeito do parâmetro η

4.5 GRASP (GM) *versus* GRASP (GMP)

A realização de religações de caminho, como estratégia de pós-otimização, mostrou-se muito eficiente. O número de instâncias de 60 tarefas que foram melhoradas chegou a 106, de um total de 120. Para as instâncias de 90 tarefas aquelas melhoradas totalizaram 116. Por fim, todas as instâncias de 120 tarefas tiveram seus custos reduzidos.

Os desvios percentuais, calculados como $\left[Z(GM) - Z(GMP)\right] / Z(\min)$, são apresentados nas Figuras 4.4 e 4.5 segundo os parâmetros τ e R, respectivamente. Desvios positivos são favoráveis a GMP. Observa-se, na Figura 4.4, que os desvios percentuais são apenas marginais nas instâncias mais apertadas (τ = 0.9). Para tais instâncias, os maiores desvios não passam de 0.08% (60 tarefas), 0.15% (90 tarefas) e 0.09% (120 tarefas), enquanto que para as instâncias mais folgadas (τ = 0.3) os maiores desvios chegam a 4.68% (60 tarefas), 4.11% (90 tarefas) e 4.83% (120 tarefas).

Em relação ao parâmetro R, os maiores desvios estão associados àquelas instâncias com R = 0.75 (4.68% para as instâncias de 60 tarefas, 4.11% para as de 90 tarefas e 4.83% para as de 120 tarefas) – ver Figura 4.5. Comportamento semelhante foi observado no item 4.4, na comparação entre GB e GM. Foi observado que o fator de severidade dos tempos de preparação não exerce influência significativa no desempenho de GMP em comparação a GM.

A variante GMP gastou cerca de 87s nas instâncias de 60 tarefas (10,63% a mais que a variante GM). Nas instâncias de 90 tarefas, o tempo médio foi de 378s (acréscimo de 23%), enquanto que para aquelas de 120 tarefas, o tempo médio foi de 1065s (aumento de 37%).

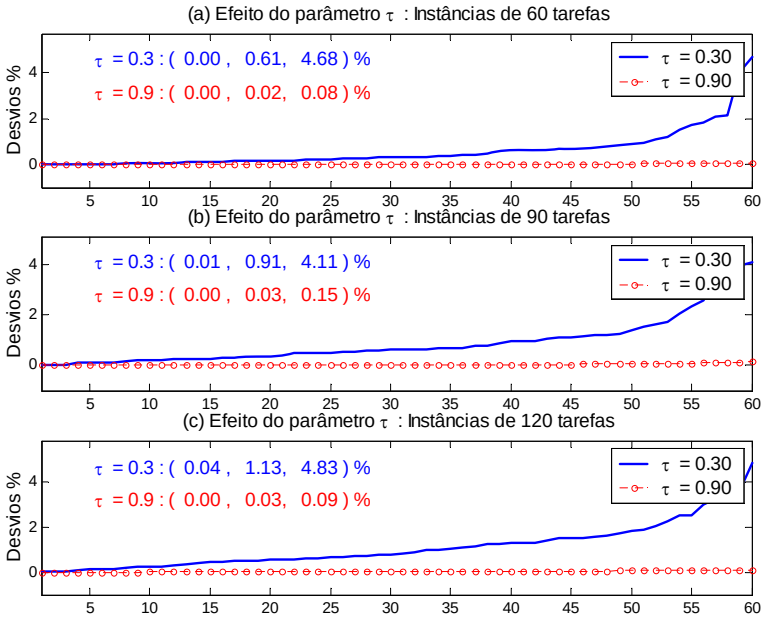

Figura 4.4 – PST: GM *versus* GMP – Efeito do parâmetro τ

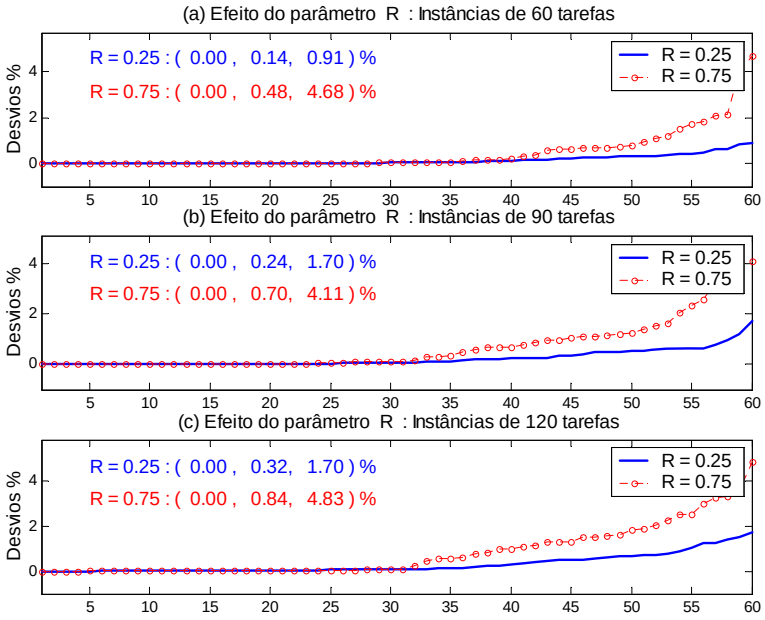

Figura 4.5 – PST: GM *versus* GMP – Efeito do parâmetro R

4.6 GRASP (GM) *versus* GRASP (GMPop)

A aplicação da estratégia POP (aqui implementada como busca local em soluções parciais) nas iterações da fase de aprendizado também possibilitou alguns melhores resultados em comparação com GM. No entanto, cabe destacar que a aplicação da estratégia POP piorou os resultados de 45 instâncias de 60 tarefas, 49 instâncias de 90 tarefas e 56 de 120 tarefas. Deve ser observado, ainda, que as melhorias são menos impactantes quando comparadas àquelas decorrentes do uso de religações de caminho (pós-otimização). Isto é, se uma escolha tiver que ser feita entre a aplicação da estratégia POP ou a aplicação de religações de caminho na pós-otimização, é preferível ficar com a segunda opção.

Nas Figuras 4.6 e 4.7 mostram-se os efeitos dos parâmetros τ e R sobre os desvios percentuais, para cada subconjunto de instâncias. Os desvios são calculados como $[Z(GM) - Z(GMPop)]/Z(\min)$. Portanto, valores positivos são favoráveis a GMPop, enquanto que valores negativos são favoráveis a GM. Mais uma vez destaca-se o fato dos desvios serem bem pequenos em instâncias com τ = 0.9 (desvio máximo de 0.27% nas instâncias de 60 tarefas, 0.33% nas de 90 tarefas e 0.23% nas de 120 tarefas). O parâmetro η não exerce influência sobre o desempenho relativo entre as duas variantes.

Os tempos de CPU gastos na variante GMPop foram de 130s (60 tarefas), 590s (90 tarefas) e 1647s (120 tarefas). Estes resultados representam aumentos de 65%, 92% e 112%, em relação aos tempos gastos pela variante GM.

O teste de Wilcoxon foi aplicado a diversos subconjuntos de instâncias, comparando GM a GMPop. As conclusões estão apresentadas na Tabela 4.4, em que é indicada a variante dominante, para cada subconjunto de instâncias. Asteriscos são utilizados nas situações em que o teste não indica a dominância de nenhuma das variantes.

Tabela 4.4 – PST: Teste de Wilcoxon (variantes dominantes) – GM *versus* GMPop

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	***	GMPop	GMPop	GMPop	***	GMPop
90	***	GMPop	***	***	GMPop	***
120	***	***	***	GMPop	***	***

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	***	GMPop	GMPop	***	GMPop
90	***	***	GMPop	GMPop	***
120	***	***	***	***	***

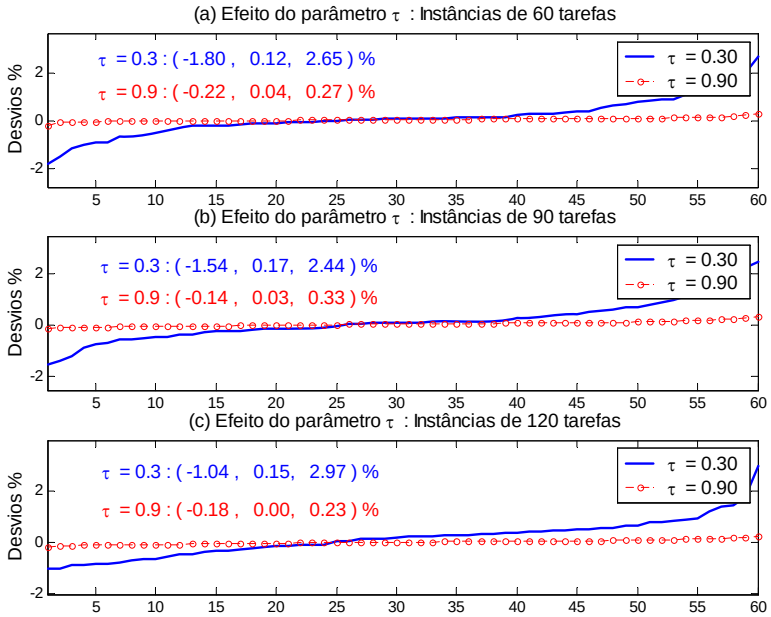

Figura 4.6 – PST: GM *versus* GMPop – Efeito do parâmetro τ

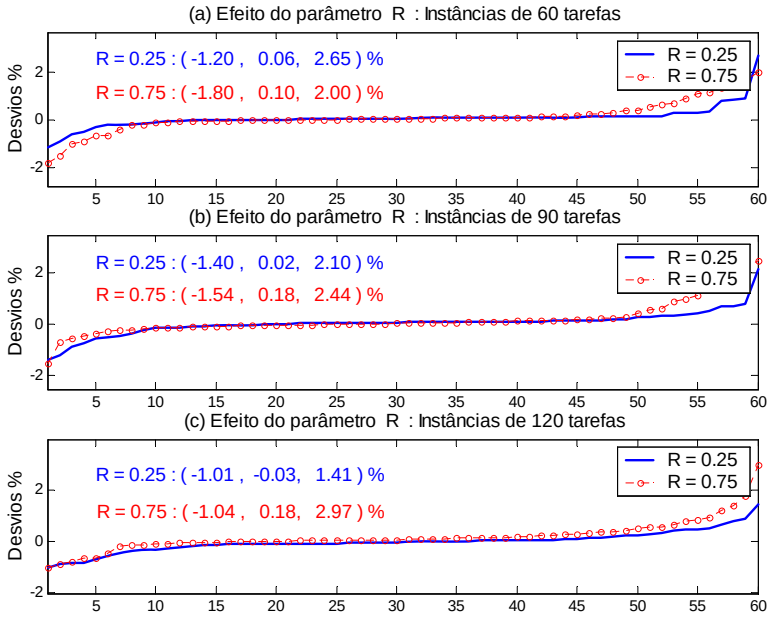

Figura 4.7 – PST: GM *versus* GMPop – Efeito do parâmetro R

4.7 GRASP (GB) *versus* Busca Tabu (TC)

O desempenho da Busca Tabu de curto prazo (TC) foi muito superior ao do GRASP Básico (GB). TC venceu GB na quase totalidade das instâncias, exceto naquelas com 12 máquinas, nas quais GB chegou a vencer em 22 instâncias de 60 tarefas (de um total de 24). Os desvios percentuais, $[Z(GB) - Z(TC)]/Z(\min)$, estão mostrados nas Figuras 4.8 e 4.9, para os parâmetros τ e R. Valores positivos são favoráveis a TC e negativos são favoráveis a GB. Na Figura 4.8 observa-se, por exemplo, que os desvios máximos (todos favoráveis a TC) foram de 0.72%, instâncias de 60 tarefas com τ = 0.9, 0.80% para as de 90 tarefas e de 0.76% para as de 120 tarefas. As conclusões do teste de Wilcoxon são apresentadas na Tabela 4.5, confirmando a superioridade de TC sobre GB, em um número significativo de instâncias.

Uma investigação das razões pelas quais o desempenho de TC foi tão inferior nas instâncias de 12 máquinas, revelou que a duração tabu, que havia sido calibrada considerando-se instâncias com 60 tarefas e 3 e 6 máquinas, estava excessiva. Experimentos adicionais, realizados sobre as instâncias de 12 máquinas, com durações tabu escolhidas aleatoriamente no intervalo $[0.0, 0.02 * n * m]$ para os movimentos *cross*, produziram uma melhora substancial no desempenho da Busca Tabu. Nos novos experimentos, TC vence GB em 21 instâncias de 60 tarefas, em 21 de 90 tarefas e em 22 de 120 tarefas. O teste de Wilcoxon passa a indicar a dominância de TC sobre GB.

Enquanto os tempos médios gastos pelo GRASP Básico foram de 72s para instâncias com 60 tarefas, 281s para as de 90 tarefas e de 715s para as de 120 tarefas, a Busca Tabu de curto prazo gastou, em média, 69s, 245s e 580s, permitindo economias de tempo computacional de 4,5%, 13% e 18,9%, respectivamente.

Tabela 4.5 – PST: Teste de Wilcoxon (variantes dominantes) : GB *versus* TC

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TC	TC	TC	TC	TC	TC
90	TC	TC	TC	TC	TC	TC
120	TC	TC	TC	TC	TC	TC

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TC	TC	TC	TC	GB(1) / TC (2)
90	TC	TC	TC	TC	*** (1) / TC (2)
120	TC	TC	TC	TC	*** (1) / TC (2)

(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6
(2) Com intervalos de duração tabu reduzidos

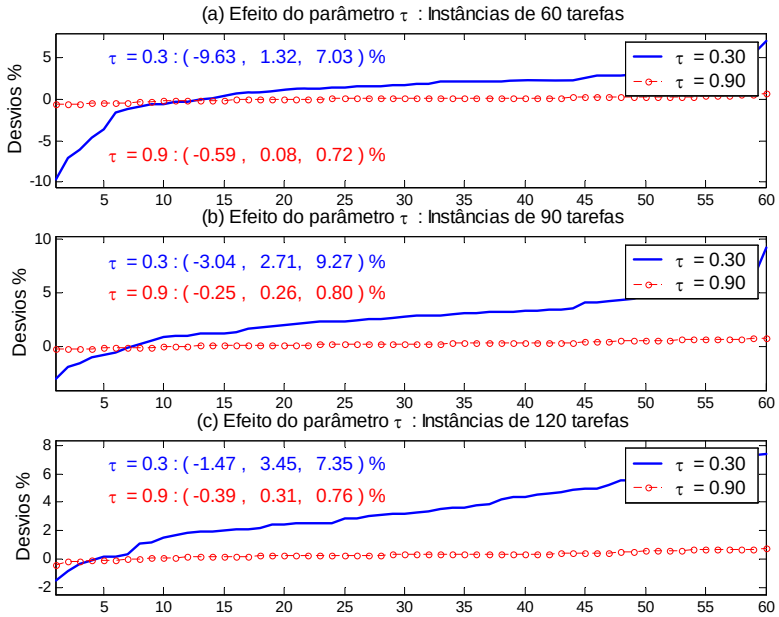

Figura 4.8 – PST: GB *versus* TC – Efeito do parâmetro τ

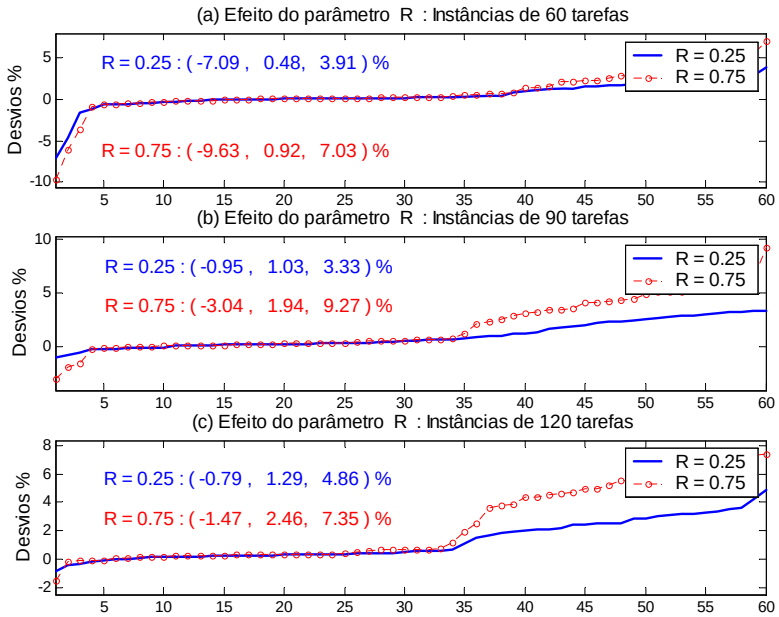

Figura 4.9 – PST: GB *versus* TC – Efeito do parâmetro R

4.8 Busca Tabu (TC) *versus* Busca Tabu (TRT)

A variante TRT (reinícios segundo Rochat e Taillard) melhorou os resultados da Busca Tabu de curto prazo (TC) em um significativo número de instâncias. Em especial, quase todas com τ = 0.3 (instâncias mais folgadas) foram melhoradas. Já em relação às instâncias mais apertadas (τ = 0.9), pouco mais da metade das instâncias foram melhoradas por TRT (38 de 60 tarefas, 39 de 90 tarefas e 34 de 120 tarefas). Não houve um único empate entre as duas variantes. Foi observado que o percentual de máquinas não preenchidas ao final da segunda fase do procedimento de Rochat e Taillard (fase de extração de seqüências) é maior nas instâncias com τ = 0.9. Também foi observado que o percentual de reinícios nos quais era obtida uma solução completa, ao final da segunda fase, com seqüências extraídas de diferentes soluções de elite, é menor em tais instâncias. Ou seja, em geral, nas instâncias com τ = 0.9, o procedimento de extração de seqüências ou preenche poucas máquinas ou preenche todas elas, mas a partir de uma única solução de elite. Estes dois fatores colaboram para uma perda de eficiência do procedimento. As análises mostraram, ainda, que quanto maior o total de máquinas, maior o percentual de máquinas não preenchidas na fase de extração de seqüências e, portanto, maior a porção da solução que precisa ser completada pela fase de factibilização. Nas Figuras 4.10 e 4.11 são mostrados os desvios percentuais entre as duas variantes. Os desvios são calculados como $\left[Z(TC) - Z(TRT)\right] / Z(\min)$, de modo que valores positivos são favoráveis a TRT. O teste de Wilcoxon, cujas conclusões são mostradas na Tabela 4.6, confirma a superioridade de TRT.

A variante TRT gastou, em média, cerca de 10% mais tempo que a Busca Tabu de curto prazo. Os tempos médios ficaram em : 72s (instâncias de 60 tarefas), 268s (90 tarefas) e 637s (120 tarefas).

Tabela 4.6 – PST: Teste de Wilcoxon (variantes dominantes) : TC *versus* TRT

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TRT	TRT	TRT	TRT	TRT	TRT
90	TRT	TRT	TRT	TRT	TRT	TRT
120	TRT	***	TRT	TRT	TRT	TRT

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TRT	TRT	TRT	TRT	TRT
90	***	TRT	TRT	TRT	TRT
120	***	TRT	TRT	TRT	TRT

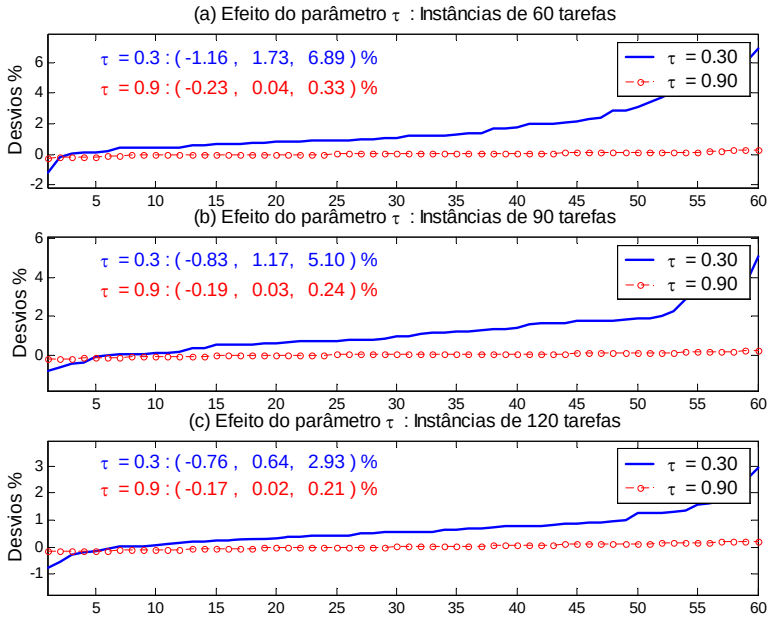

Figura 4.10 – PST: TC *versus* TRT – Efeito do parâmetro τ

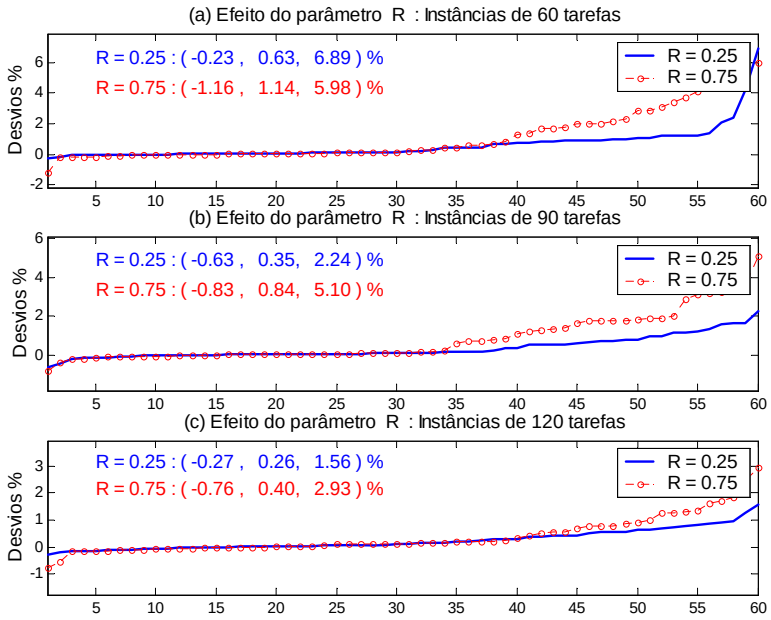

Figura 4.11 – PST: TC *versus* TRT – Efeito do parâmetro R

4.9 Busca Tabu (TC) *versus* Busca Tabu (TDiv)

Foram realizados diversos testes variando os parâmetros de controle da variante TDiv (freqüência de ativação da fase de diversificação, duração da fase de diversificação, grau de penalização). De um modo geral, os resultados de TDiv foram ruins (piores que a Busca Tabu de curto prazo), confirmando, por um lado, que a estratégia cumpriu seu papel ao conduzir a busca para regiões que de outro modo não seriam exploradas e, por outro lado, a necessidade de uma estratégia de intensificação.

4.10 Busca Tabu (TC) *versus* Busca Tabu (TDivRC)

A realização de religações intermediárias, após a obtenção de um mínimo local ao final da fase de diversificação, fez com que os resultados de TDiv ficassem muito melhores, superando a Busca Tabu de curto prazo em 105 instâncias de 60 tarefas, em 90 instâncias de 90 tarefas e em 76 instâncias de 120 tarefas. A superioridade de TDivRC sobre TC pode ser vista nas Figuras 4.12 e 4.13, que mostram os desvios percentuais calculados como $\left[Z(TC) - Z(TDivRC)\right] / Z(\min)$ (valores positivos são, portanto, favoráveis a TDivRC), bem como na Tabela 4.7, em que são mostradas as conclusões do teste de Wilcoxon.

A variante TDivRC gastou, em média, cerca de 18% mais tempo que a Busca Tabu de curto prazo, para as instâncias de 60 e 90 tarefas e cerca de 20% a mais para as instâncias de 120 tarefas. Os tempos médios ficaram em : 81s (instâncias de 60 tarefas), 288s (90 tarefas) e 697s (120 tarefas).

Tabela 4.7 – PST: Teste de Wilcoxon (variantes dominantes) : TC *versus* TDivRC

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC
90	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC
120	TDivRC	TDivRC	TDivRC	***	TDivRC	TDivRC

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	***	TDivRC	TDivRC	TDivRC	TDivRC
90	***	TDivRC	TDivRC	TDivRC	TDivRC
120	***	***	***	TDivRC	TDivRC

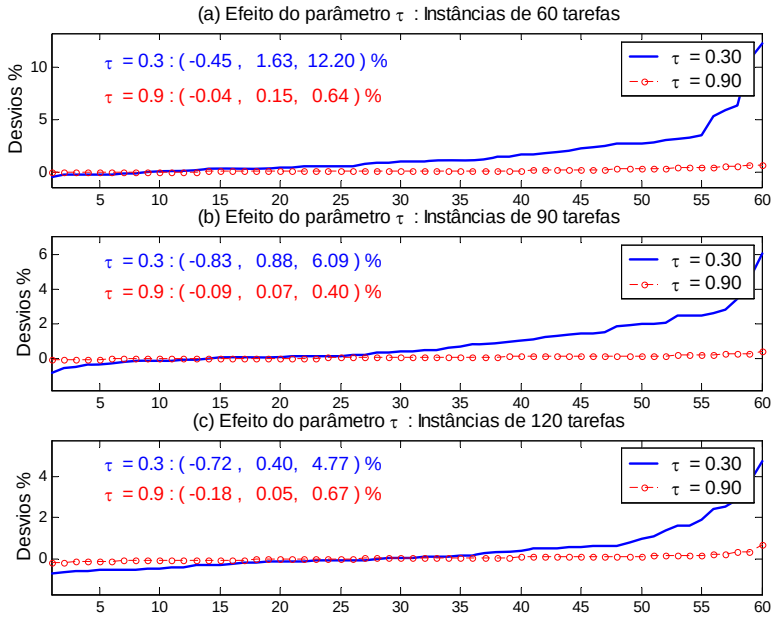

Figura 4.12 – PST: TC *versus* TDivRC – Efeito do parâmetro τ

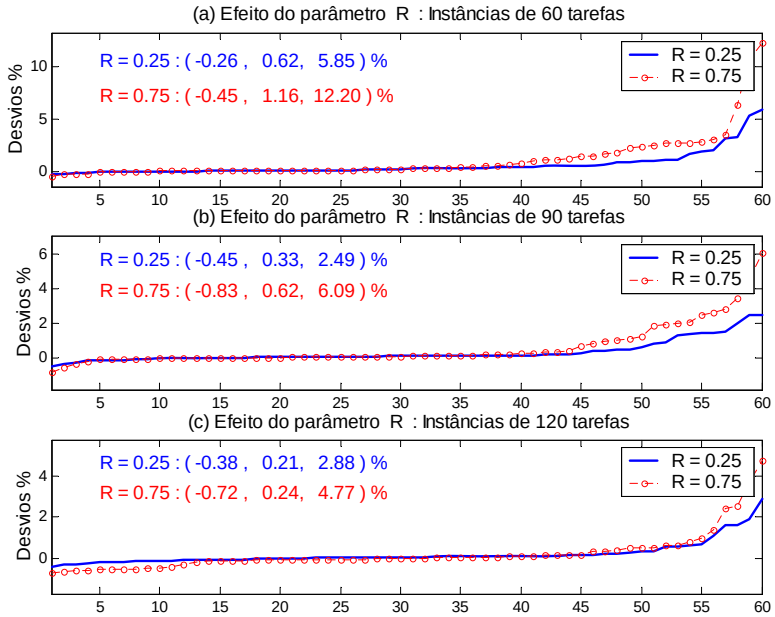

Figura 4.13 – PST: TC *versus* TDivRC – Efeito do parâmetro R

4.11 Busca Tabu (TDivRC) *versus* Busca Tabu (TRC)

A Busca Tabu com fase de diversificação e religações de caminho intermediárias (TDivRC) apresentou desempenho muitíssimo semelhante à Busca Tabu sem diversificação mas com religações de caminho intermediárias (TRC). Os desvios percentuais, calculados como $\left[Z(TDivRC) - Z(TRC)\right] / Z(\min)$ apresentam médias próximas de zero, para todos os subconjuntos de instâncias de mesmo número de tarefas. Os desvios favoráveis a TDivRC chegam a –4.89% nas instâncias de 60 tarefas, –4.15% (90 tarefas) e –5.71% (120 tarefas), enquanto que os desvios favoráveis a TRC chegam a 3.78% (60 tarefas), 2.90% (90 tarefas) e 2.87% (120 tarefas). De acordo com o teste de Wilcoxon, uma variante não domina a outra. Para as instâncias avaliadas, as religações de caminho é que são determinantes para a qualidade final das soluções encontradas.

4.12 Busca Tabu (TRT) *versus* Busca Tabu (TDivRC)

A Busca Tabu com reinícios segundo Rochat e Taillard (TRT) saiu-se melhor que a Busca Tabu com diversificação baseada em freqüência de residência e religações intermediárias (TDivRC) nas instâncias mais folgadas em relação à média das datas de entrega (τ = 0,3) e nas instâncias mais apertadas em termos da razão entre o número de tarefas e o de máquinas. A situação fica invertida, ou seja, favorável a TDivRC, quando são consideradas as instâncias com médias das datas de entrega mais apertadas (τ = 0,9) e com menor número de tarefas por máquina. De um modo geral, no entanto, a variante TRT saiu-se melhor que TDivRC. Tais resultados são apresentados nas Figuras 4.14 e 4.15, bem como na Tabela 4.8 (teste de Wilcoxon). Os desvios são calculados como $\left[Z(TRT) - Z(TDivRC)\right] / Z(\min)$. Valores positivos são favoráveis a TDivRC. A variante TDivRC venceu a variante TRT em 67 instâncias de 60 tarefas (de um total de 120), em 57 de 90 tarefas e em 48 de 120 tarefas. Não houve empates.

Na comparação entre os tempos de CPU das variantes TRT e TDivRC, esta última gastou, em média, 7.5% mais tempo que a primeira, nas instâncias de 60 e 90 tarefas e cerca de 9.4% a mais nas instâncias de 120 tarefas.

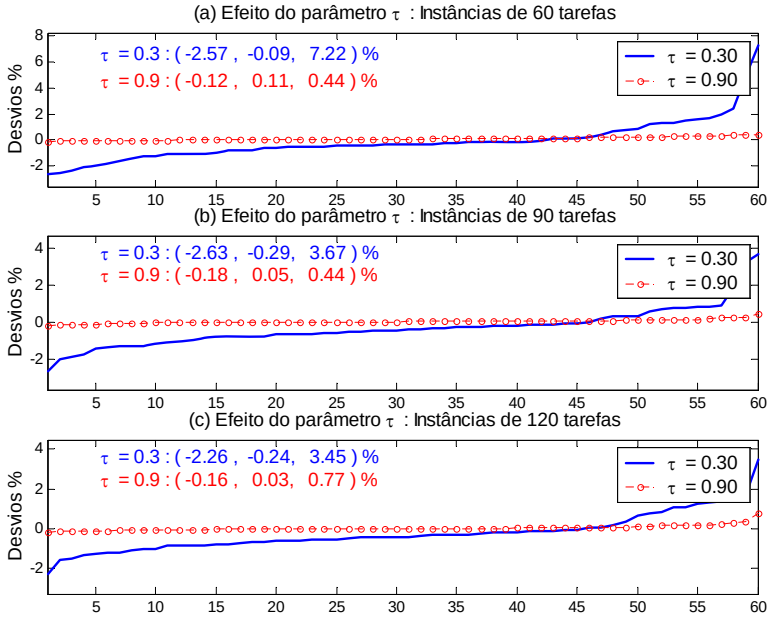

Figura 4.14 – PST: TRT *versus* TDivRC – Efeito do parâmetro τ

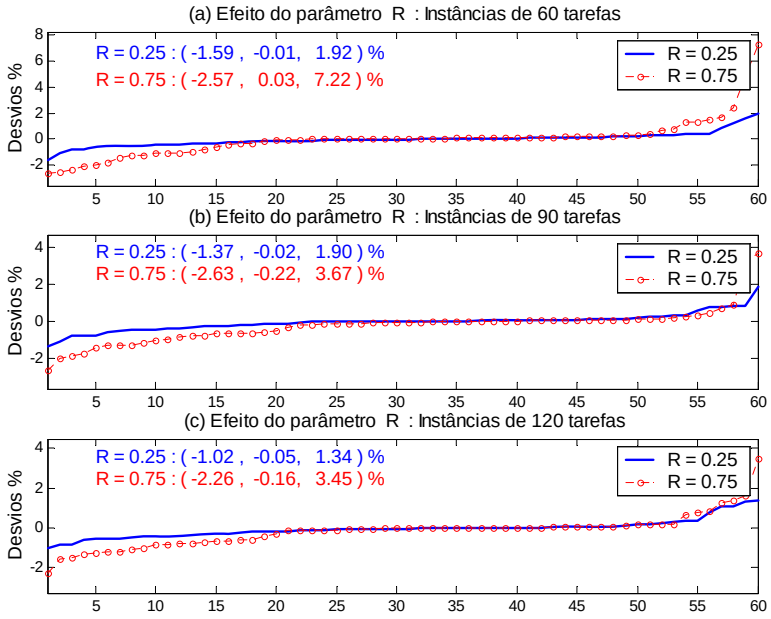

Figura 4.15 – PST: TRT *versus* TDivRC – Efeito do parâmetro R

Tabela 4.8 – PST: Teste de Wilcoxon (variantes dominantes) : TRT *versus* TDivRC						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TRT	TDivRC	***	***	***	***
90	TRT	TDivRC	***	TRT	TRT	***
120	TRT	***	TRT	TRT	TRT	***

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TRT	TRT	***	***	TDivRC
90	TRT	TRT	TRT	***	TDivRC
120	***	TRT	TRT	TRT	TDivRC

4.13 GRASP (GMP) *versus* Busca Tabu (TRTP)

Para os subconjuntos de instâncias com τ = 0.3 e com até 6 máquinas, a Busca Tabu venceu o GRASP na quase totalidade das instâncias. Considerando tanto as instâncias com τ = 0.3, quanto aquelas com τ = 0.9, TRTP saiu-se melhor que GMP em 71 instâncias de 60 tarefas, em 75 de 90 tarefas e em 61 de 120 tarefas. Não houve empates. Os desvios, dados por $\left[Z(GMP) - Z(TRTP)\right] / Z(\min)$, são mostrados nas Figuras 4.16 e 4.17, para os parâmetros τ e R. Já as conclusões do teste de Wilcoxon são apresentadas na Tabela 4.9, em que se observa, dentre outros, o domínio de GMP nas instâncias com 12 máquinas, quando os intervalos de duração tabu são aqueles determinados nos testes preliminares (item 4.3). Como observado na comparação entre o GRASP básico e a Busca Tabu de curto prazo, o pior desempenho de TRTP, nos conjuntos de instâncias com 12 máquinas, é fruto da duração tabu, que foi excessiva. Testes adicionais, nos quais a duração tabu para os movimentos *cross* é escolhida no intervalo $[0.0, 0.02 * n * m]$ indicam uma melhora de TRTP, a ponto dela não mais ser dominada por GMP. Já o desempenho pior nas instâncias mais apertadas, deve-se aos motivos apresentados no item 4.9.

Os tempos de GMP foram, em média, 87s, 378s e 1065s para as instâncias de 60, 90 e 120 tarefas, enquanto que os tempos de TRTP ficaram em 88s, 345s e 883s.

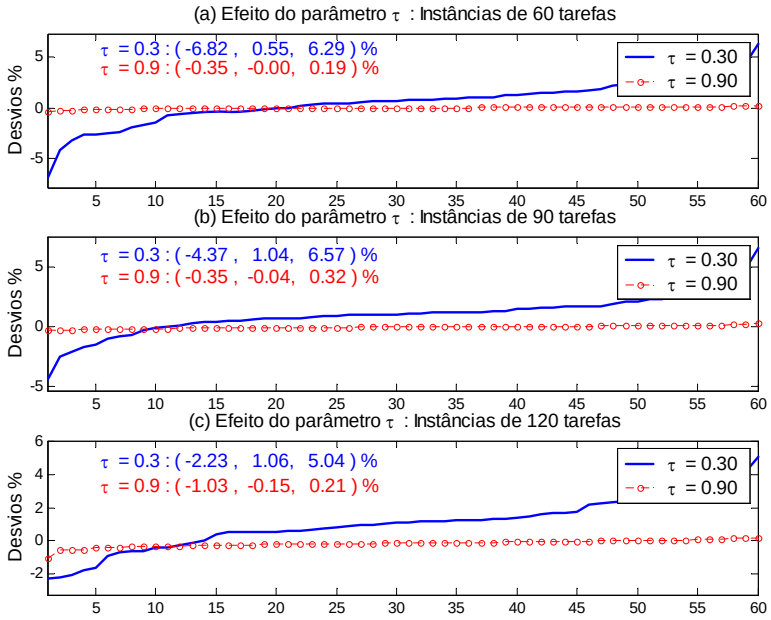

Figura 4.16 – PST: GMP *versus* TRTP – Efeito do parâmetro τ

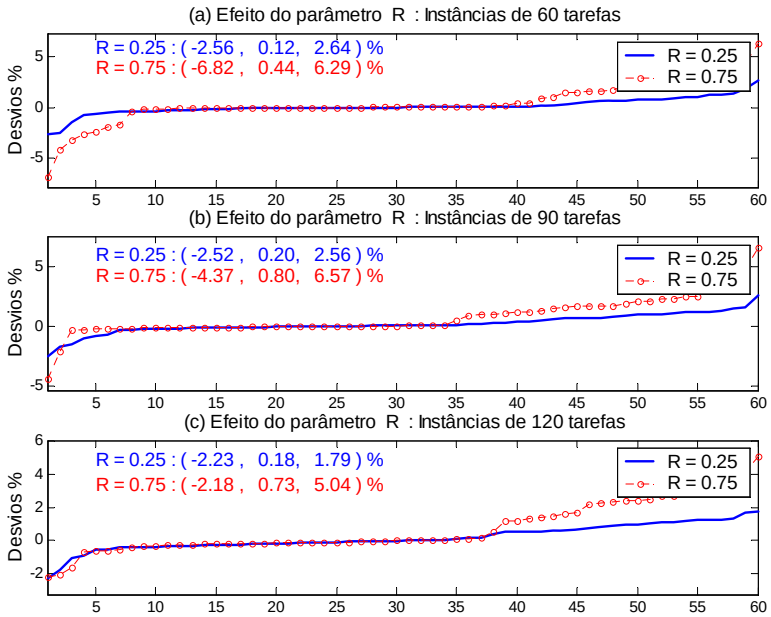

Figura 4.17 – PST: GMP *versus* TRTP – Efeito do parâmetro R

Tabela 4.9 – PST: Teste de Wilcoxon (variantes dominantes) : GMP *versus* TRTP						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TRTP	***	***	TRTP	TRTP	***
90	TRTP	GMP	TRTP	TRTP	TRTP	TRTP
120	TRTP	GMP	***	TRTP	TRTP	***
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	TRTP	TRTP	TRTP	***	GMP (1) / *** (2)	
90	TRTP	TRTP	TRTP	TRTP	GMP (1) / *** (2)	
120	TRTP	TRTP	TRTP	***	GMP (1) / *** (2)	
(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6 (2) Com intervalos de duração tabu reduzidos						

4.14 GRASP (GMP) *versus* Busca Tabu (TDivRCP)

De modo semelhante àquele observado no item 4.13, o desempenho da Busca Tabu de longo prazo (TDivRCP) foi melhor que o do GRASP com memória acrescido de religações de caminho (GMP). Os totais de vitórias da Busca Tabu foram mais expressivos para as instâncias com τ = 0.3 e com até 6 máquinas. Os desvios entre GMP e TDivRCP estão mostrados nas Figuras 4.18 e 4.19, em termos dos parâmetros τ e R. Valores positivos são favoráveis à variante de Busca Tabu. As conclusões do teste de Wilcoxon são mostradas na Tabela 4.10. O desempenho de TDivRCP, nas instâncias de 12 máquinas, ficou prejudicado pela duração tabu pouco apropriada. Testes adicionais para tais instâncias, empregando durações tabu menores mostram a melhora do desempenho de TDivRCP.

Os tempos de GMP foram, em média, 87s, 378s e 1065s para as instâncias de 60, 90 e 120 tarefas, enquanto que od de TDivRCP chegaram a 95s, 371s e 966s.

Tabela 4.10 – PST: Teste de Wilcoxon (variantes dominantes) : GMP *versus* TDivRCP						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TDivRCP	TDivRCP	TDivRCP	TDivRCP	TDivRCP	TDivRCP
90	TDivRCP	***	TDivRCP	TDivRCP	TDivRCP	***
120	TDivRCP	GMP	***	TDivRCP	TDivRCP	***
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	TDivRCP	TDivRCP	TDivRCP	***	GMP (1) / *** (2)	
90	TDivRCP	TDivRCP	TDivRCP	TDivRCP	GMP (1) / *** (2)	
120	TDivRCP	TDivRCP	TDivRCP	***	GMP (1) / *** (2)	
(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6 (2) Com intervalos de duração tabu reduzidos						

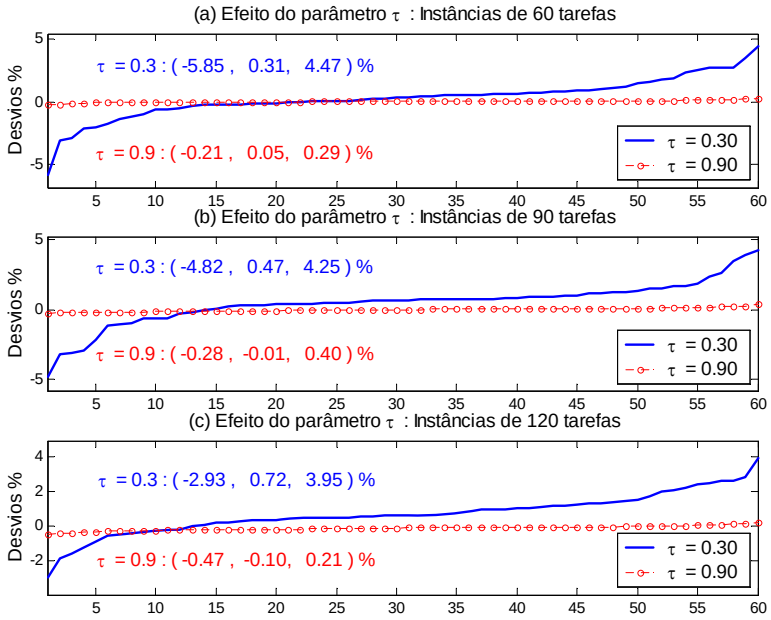

Figura 4.18 – PST: GMP *versus* TDivRCP – Efeito do parâmetro τ

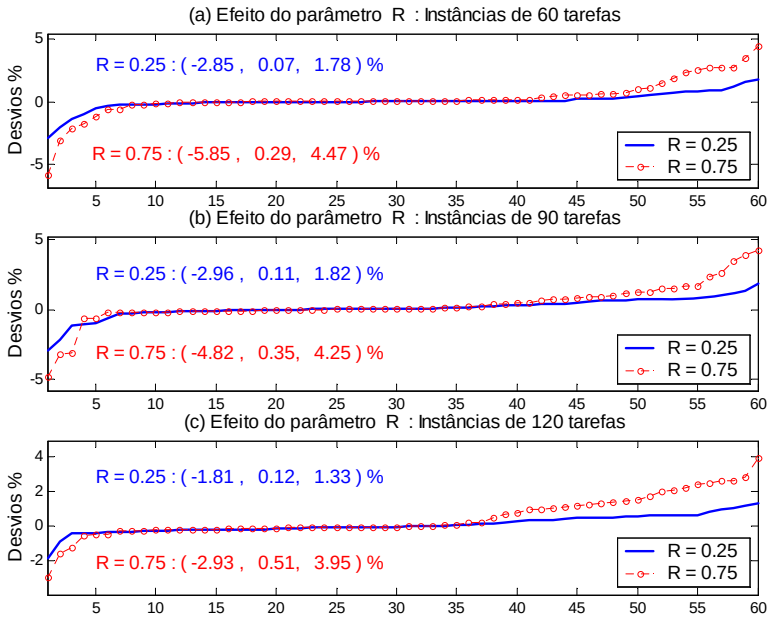

Figura 4.19 – PST: GMP *versus* TDivRCP – Efeito do parâmetro R

4.15 Médias dos tempos de CPU – Resumo

As médias dos tempos de CPU (em segundos) gastas pelas diversas variantes implementadas são apresentadas nas Figuras 4.20, 4.21 e 4.22, para as instâncias de 60, 90 e 120 tarefas, respectivamente. Como mostrado ao longo das comparações apresentadas nos itens anteriores, as estratégias implementadas objetivando melhorias nos desempenhos das versões básicas, tanto do GRASP, quanto da Busca Tabu, impõem acréscimos nas médias dos tempos de CPU, que são significativamente compensados pelos desempenhos superiores. A única exceção está associada à estratégia POP, cuja relação custo-benefício não se mostrou satisfatória.

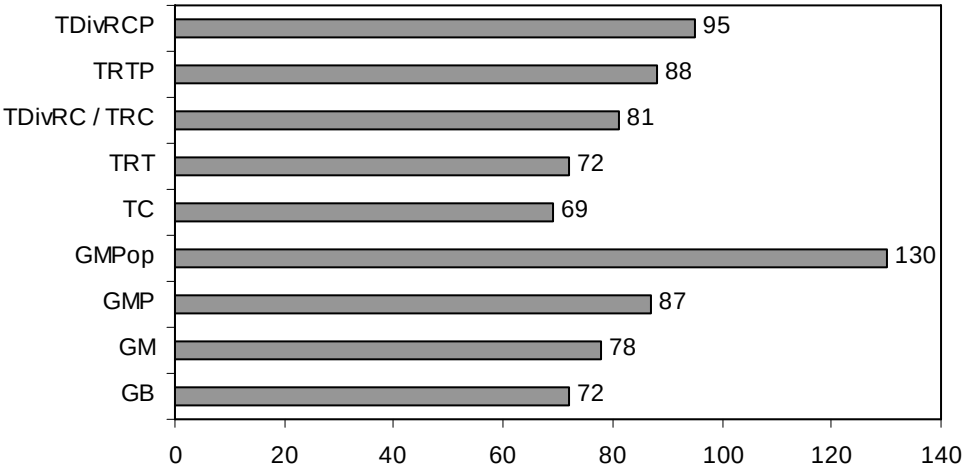

Figura 4.20 – PST: Médias dos tempos de CPU (s) para as instâncias de 60 tarefas

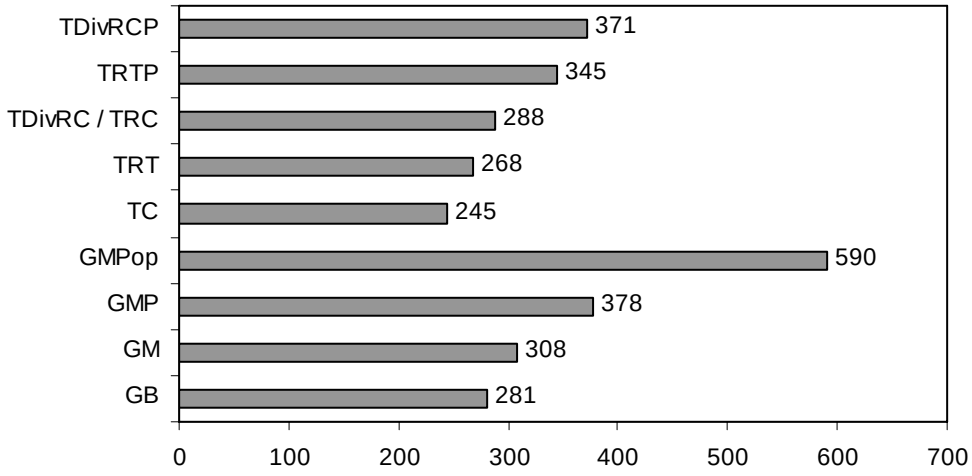

Figura 4.21 – PST: Médias dos tempos de CPU (s) para as instâncias de 90 tarefas

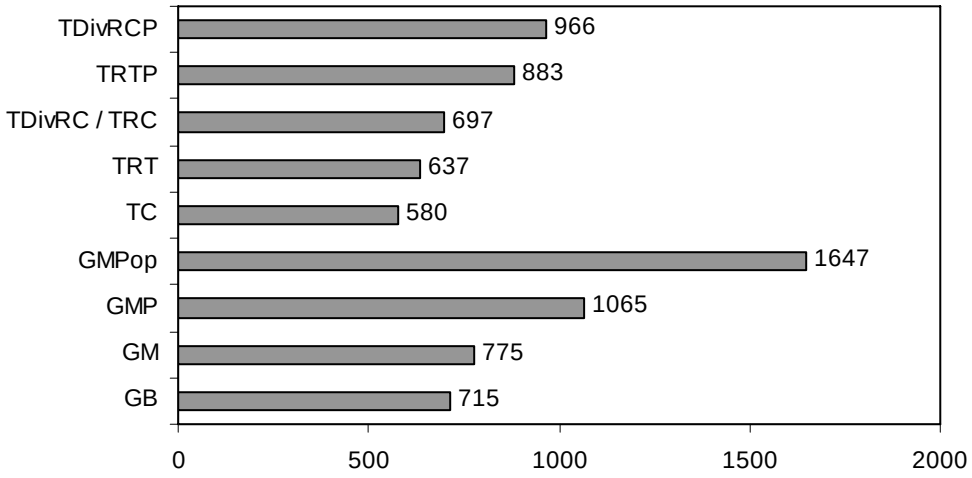

Figura 4.22 – PST: Médias dos tempos de CPU (s) para as instâncias de 120 tarefas

Capítulo 5

Testes computacionais: Problema PSET

5.1 Instâncias e critério de parada

Foram geradas 360 instâncias, com o mesmo procedimento adotado para o problema **PST**. Há 24 instâncias para cada combinação de número de máquinas, $m = (2,3,4,6,12)$, e tarefas, $n = (60,90,120)$. As duas únicas diferenças em relação ao problema **PST** são: i) instante de liberação de cada tarefa, r_i, é definido no intervalo $\left[0,\ \max(0,\ d_i - \bar{s} - \bar{p})\right]$ e ii) penalizações por unidade de tempo de adiantamento são distribuídas com probabilidade uniforme no intervalo $[1,100]$, e são independentes das penalizações por atraso. Deve ser observado que não há garantia de que $r_i < d_i$ para toda e qualquer tarefa *i*.

O critério de parada é estabelecido em termos de um número máximo de soluções investigadas, que depende da soma dos tamanhos das vizinhanças de troca e de inserção.

Os códigos foram implementados em linguagem C++ e foram compilados no Microsoft Visual C++ 6.0. Todos os testes foram realizados em um computador com processador AMD Athlon de 1.0 GHz, com 512MB de memória RAM.

5.2 Caracterização das variantes e testes preliminares

Os testes preliminares tiveram por objetivo a calibração dos parâmetros de controle presentes em cada variante das meta-heurísticas GRASP e Busca Tabu. Também serviram para balizar a escolha dos tipos de movimentos (trocas e inserções) *versus* (*cross* e *Or-Opt*). Foram conduzidos sobre um conjunto reduzido de instâncias (aquelas de 60 tarefas, com 3 e 6 máquinas).

GRASP Básico (GB)

- Função gulosa: Baseada no índice MATCS (Equação 3.9);
- Parâmetro α (Equação 3.11): Escolhido aleatoriamente com probabilidade uniforme no intervalo $[0.1, 0.5]$, por ter propiciado melhores resultados para o problema **PST**;
- Tarefas da LRC: Escolhidas com probabilidade diretamente proporcional ao valor da função gulosa (Equação 3.12);
- Busca local: Foi constatada uma superioridade da variante com movimentos de troca e inserção sobre a variante com movimentos *cross* e *Or-Opt*;
- Parâmetro ξ de redução de vizinhança: Foram avaliados os valores de ξ = 0.0 (vizinhança completa), 0.3, 0.6 e 0.9. Os melhores resultados foram alcançados utilizando-se ξ = 0.6.

GRASP com Memória (GM)

- Função gulosa, parâmetro α, parâmetro ξ de redução de vizinhança (idênticos aos utilizados em **GB**);
- Busca local: Vizinhanças alternadas de troca e de inserção;
- Duração da fase de aprendizado: 20% do total de soluções a serem investigadas;
- Função de intensidade: Calculada segundo Equação 3.13;
- Tarefas da LRC: Escolhidas com probabilidade diretamente proporcional ao valor da função gulosa (Equação 3.17);
- Parâmetro λ para ponderação entre a função gulosa e a função de intensidade: Definido a partir da função de entropia (Equação 2.12), com valores no intervalo 0.1 a 0.9 e incrementos de 0.1. A função de entropia é reavaliada a cada x_1 = 10 iterações;
- Cardinalidade do conjunto de elite, $|\Gamma|$: Foi adotado o valor 10 (que forneceu os melhores resultados para o problema **PST**);
- Limiar de distância, $D_{\min}$, entre soluções de elite: Foram avaliados os valores 0.3, 0.5 e 0.7. Assim como no problema **PST**, não foram observados desempenhos muito diferentes entre eles. Optou-se, dessa forma, pelo valor intermediário, $D_{\min}$ = 0.5;
- Parâmetro θ, que limita o fator de escala do critério de aceitação de soluções no conjunto de elite: Foram avaliados os valores θ = 1.0 e θ = 0.5. Os resultados alcançados com θ = 0.5 foram ligeiramente melhores.

GM + Religações de caminhos na pós-otimização (GMP)

- Trajetórias de ligação: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: inserção (variante RC1);
- Buscas locais ao longo das trajetórias: Vizinhanças de troca e de inserção, exploradas alternadamente;
- Demais parâmetros assumem os valores estabelecidos para **GM**.

GM + POP na fase de aprendizado (GMPop)

- Ponto de aplicação da busca local sobre a solução parcial: Ao alocar 75% das tarefas;
- Buscas locais sobre a solução parcial: Vizinhanças de troca e de inserção;
- POP aplicado apenas nas iterações da fase de aprendizado;
- Demais parâmetros assumem os valores estabelecidos para **GM**.

GMPOP + Religações de caminhos na pós-otimização (GMPP)

- Parâmetros assumem valores definidos para as variantes **GM**, **GMP** e **GMPop**.

Busca Tabu de Curto Prazo (TC)

- Solução inicial: Heurística construtiva baseada em inícios preferenciais (item 3.4.1);
- Buscas locais e transição entre vizinhanças: Foi observado um melhor desempenho da variante baseada em movimentos de troca e de inserção (executados alternadamente). Movimentos entre máquinas (tanto trocas quanto inserções) dão origem a vizinhanças de tamanhos proporcionais a $n * m$. Por outro lado, movimentos em uma mesma máquina (tanto trocas quanto inserções) geram vizinhanças cujos tamanhos são proporcionais a $n \div m$. Assim, foi definido que a transição dos movimentos entre máquinas para os movimentos internos (uma vez detectata a estagnação local da busca) deveria ocorrer após um submúltiplo do produto $n * m$, enquanto que a transição em sentido contrário deveria ocorrer após um submúltiplo da razão $n \div m$. Testes realizados para o problema **PST** indicaram os melhores resultados associados à transição de um movimento externo para um interno após $n * m \div 60$ iterações sem melhoria na solução de partida e associados à transição em sentido contrário após $n \div m \div 5$ iterações com estagnação (em cada máquina);

- Parâmetro ξ de redução de vizinhança: Foi utilizado o valor 0.6 (mesmo do GRASP);
- Critérios de proibição e duração tabu: Os testes realizados indicaram como melhor estratégia a adoção do critério de proibição C_{out}, quando da realização de movimentos internos e do critério C_{atrib}, quando da realização de movimentos externos. O critério C_{out} proíbe movimentos que retirem arcos recém adicionados à solução, enquanto que o critério C_{atrib} proíbe o retorno de uma tarefa para qualquer máquina nas quais ela tenha estado em um passado recente. Para os movimentos entre máquinas, o intervalo de duração tabu mais adequado foi $[0.0, 0.75 * n \div m]$. Para os movimentos internos, o intervalo mais adequado foi $[0.0, 0.15 * n * m]$;
- Critério de aspiração: Um movimento tabu é admitido se permitir melhorar a solução incumbente.

TC + Reinícios segundo Rochat&Taillard (TRT)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo: Ocorre após um certo número de alternâncias entre movimentos de troca e de inserção (ciclos de curto prazo), sem melhoria da solução incumbente. Como observado para o problema **PST**, instâncias mais folgadas, tanto em termos do fator de aperto das datas de entrega (τ), quanto em termos da relação tarefas/máquinas (μ) são beneficiadas quando a estratégia de longo prazo é ativada mais freqüentemente. Desse modo, a ativação da estratégia de longo prazo foi definida para ocorrer após $1.5.\tau.\mu$ ciclos de curto prazo com estagnação da incumbente;
- A cardinalidade do conjunto de elite (utilizado para a construção do banco de sequências de tarefas alocadas a uma mesma máquina) foi fixada em 10 e o limiar de distância ($D_{\min}$) foi fixado em 0.5 (mesmos valores utilizados em **GM**).

TRT + Religações de caminhos na pós-otimização (TRTP)

- Trajetórias de ligação na pós-otimização: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: inserção;
- Buscas locais ao longo das trajetórias: Vizinhanças de troca e de inserção;
- Demais parâmetros assumem os valores estabelecidos para a variante **TRT**.

TC + Diversificação por frequência de residência (TDiv)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo: Segue critério adotado para a variante **TRT**;
- Duração da fase de diversificação e intensidade da penalização (fator γ): A duração da fase de diversificação é definida em termos de números de ciclos de alternância entre movimentos de troca e movimentos de inserção. Foram realizados diversos testes, com durações de 2, 4, 6 e 8 ciclos. Para cada duração da fase de diversificação o fator γ assumiu os valores de 0.1 a 1.0, com incrementos de 0.1. Os melhores resultados foram alcançados com a duração de 2 ciclos e $\gamma = 0.2$.

TDiv + Religações de caminhos intermediárias (TDivRC)

- Fases de curto prazo e de diversificação com os mesmos parâmetros empregados na variante **TDiv**;
- Solução de elite de referência para a realização da religação de caminho: a mais distante em relação à solução de mínimo local obtida após o término da fase de diversificação. O objetivo é explorar caminhos mais longos;
- Trajetórias de ligação: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: inserção;
- Buscas locais ao longo das trajetórias: Vizinhanças de troca e de inserção.

TC + Religações de caminhos intermediárias (TRC)

- Os ciclos de curto prazo são iguais àqueles da variante **TC**;
- Ativação da estratégia de longo prazo: Segue o mesmo critério adotado nas variantes **TRT** e **TDiv**;
- Solução de elite de referência para a realização da religação de caminho: a mais distante em relação à solução atual;
- Trajetórias de ligação: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: inserção;
- Buscas locais ao longo das trajetórias: Vizinhanças de troca e de inserção.

TRC + Religações de caminhos na pós-otimização (TRCP)

- Trajetórias de ligação na pós-otimização: Regressiva e Progressiva;
- Movimentos entre máquinas distintas, ao longo das trajetórias: inserção;
- Buscas locais ao longo das trajetórias: Vizinhanças de troca e de inserção;
- Demais parâmetros assumem os valores estabelecidos para a variante **TRC**.

O resultado mais importante da fase de testes preliminares foi o desempenho superior alcançado com a utilização de movimentos de troca e de inserção, em comparação àquele obtido com os movimentos *cross* e *Or-Opt*, tanto no GRASP quanto na Busca Tabu. Este resultado é oposto ao verificado no problema **PST**. Deve ser observado que um mesmo movimento aplicado a soluções idênticas (mesmas atribuições e mesmos seqüenciamentos) para os dois problemas induzem variações de custos menos acentuadas no problema **PST**. Assim, no problema **PST**, ao se investigar movimentos que promovem alterações mais profundas na estrutura da solução, consegue-se visitar, de maneira mais eficiente, regiões do espaço com características bem diferentes, mas que contém soluções de boa qualidade. Já no problema **PSET**, pequenas alterações na solução podem produzir grandes variações de custo. Logo, é necessário explorar o espaço de maneira mais meticulosa.

5.3 GRASP (GB) *versus* GRASP (GM)

O GRASP com Memória (GM) vence o GRASP Básico (GB) em 100 instâncias de 60 tarefas, em 106 de 90 tarefas e em 106 de 120 tarefas. A aplicação do teste de Wilcoxon revela que GM é melhor que GB, quaisquer que sejam os subconjuntos de instâncias considerados (instâncias de mesmas dimensões, instâncias com mesmo número de tarefas e mesmo parâmetro τ, etc.).

Os desvios percentuais, dados por $\left[Z(GB) - Z(GM)\right] / Z(\min)$ são apresentados na Figura 5.1(a) para as instâncias de 60 tarefas. No cálculo do desvio, $Z(GB)$ representa o custo obtido pelo GRASP básico, $Z(GM)$ representa o custo obtido pelo GRASP com memória e $Z(\mathbf{min})$ representa o menor custo conhecido para cada instância. Desvios positivos indicam

vitórias de GM sobre GB. As instâncias estão agrupadas em função do parâmetro τ e ordenadas em função de valores crescentes dos desvios percentuais. Desvios mais expressivos ocorrem em instâncias mais folgadas (τ = 0.3). Para estas instâncias, o menor desvio foi de –0.52% (valor negativo é favorável a GB). A média dos desvios foi de 0.28%, enquanto que o desvio máximo foi de 3.06%. Para as instâncias com τ = 0.9, o menor desvio observado foi de –0.06%. Na média, a superioridade de GM se traduziu em um desvio de 0.23%, com o maior valor alcançando 0.92%. Para as instâncias de 90 tarefas, com τ = 0.3, Figura 5.1(b), os desvios mínimo, médio e máximo foram de –0.68%, 0.36% e 2.36%, respectivamente. Também na Figura 5.1(b), observa-se, para as instâncias com τ = 0.9, que os desvios mínimo, médio e máximo ficaram em 0.01%, 0.37% e 1.06%. Finalmente, na Figura 5.1(c) tem-se os desvios para as instâncias de 120 tarefas, cujos valores variam de –1.33% até 2.93%, nas instâncias com τ = 0.3 e de 0.09% a 1.88% nas demais instâncias (τ = 0.9).

Os desvios favoráveis ao GRASP com memória são um pouco mais expressivos nas instâncias em que as datas de entrega são mais espalhadas em torno da data de entrega média, como mostrado na Figura 5.2. Para as instâncias de 60 tarefas e datas de entrega menos espalhadas, R = 0.25, os desvios mínimo, médio e máximo foram de –0.14%, 0.24% e 0.97%. Nas demais (R = 0.75), os desvios foram de –0.52%, 0.27% e 3.06%. Para o conjunto de instâncias de 90 tarefas, os desvios variam de –0.32% a 1.65% (R = 0.25) e de –0.68% a 2.36% (R = 0.75). Nas instâncias de 120 tarefas, com R = 0.25, a variação é de –0.43% a 1.88%, e de –1.33% a 2.93% para as instâncias com R = 0.75.

Na Figura 5.3 são apresentados os desvios quando as instâncias são agrupadas em termos do fator de severidade dos tempos de preparação (fator η). Observam-se maiores médias de desvios, favoráveis a GM, associadas às instâncias com η = 0.75.

O valor médio dos tempos de CPU gastos pelo GRASP Básico para as instâncias de 60 tarefas foi de 83s, enquanto que o GRASP com Memória, gastou, em média, 88s. Um acréscimo médio de 7%, para avaliar o mesmo número de soluções. Para as instâncias de 90 tarefas, GB gastou, em média, 360s, contra 389s de GM (aumento de 8%). Para as instâncias de 120 tarefas, o tempo médio gasto por GB foi de 1015s, enquanto que GM gastou 1115s (aumento de 9,9%).

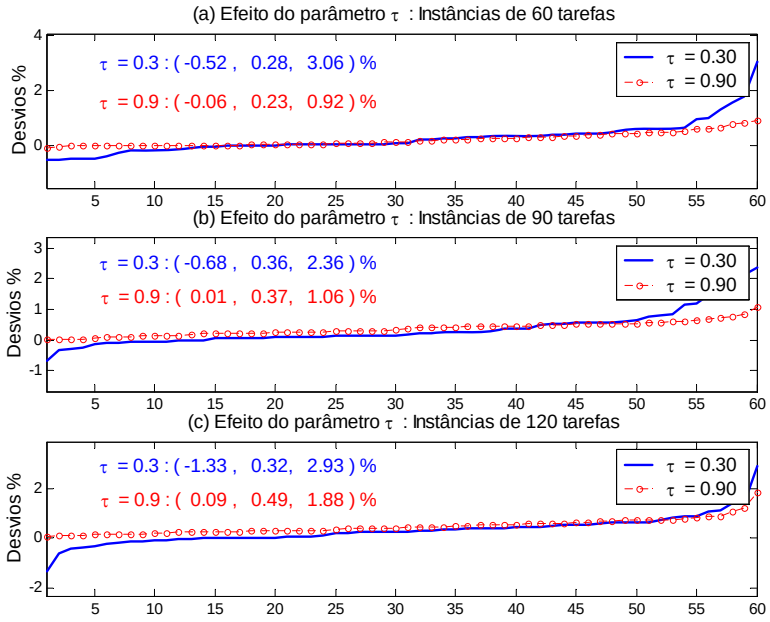

Figura 5.1 – PSET: GB *versus* GM – Efeito do parâmetro τ

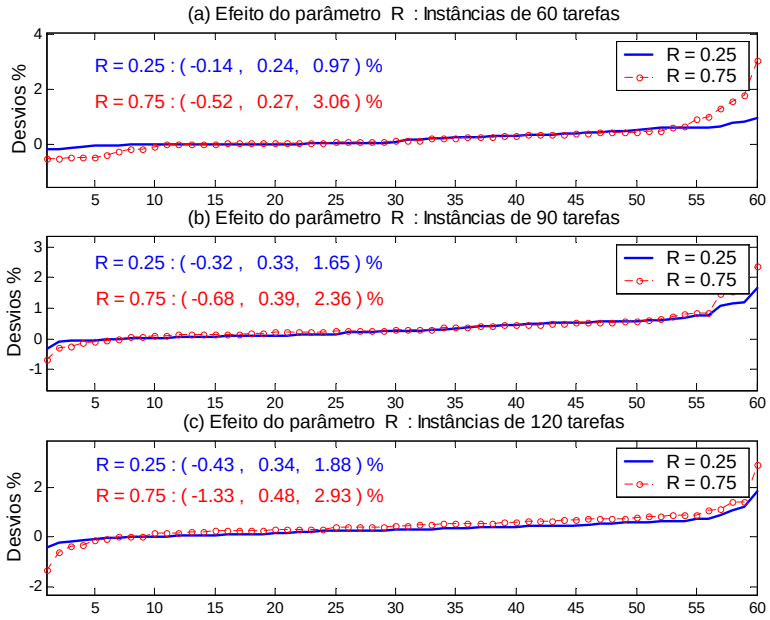

Figura 5.2 – PSET: GB *versus* GM – Efeito do parâmetro R

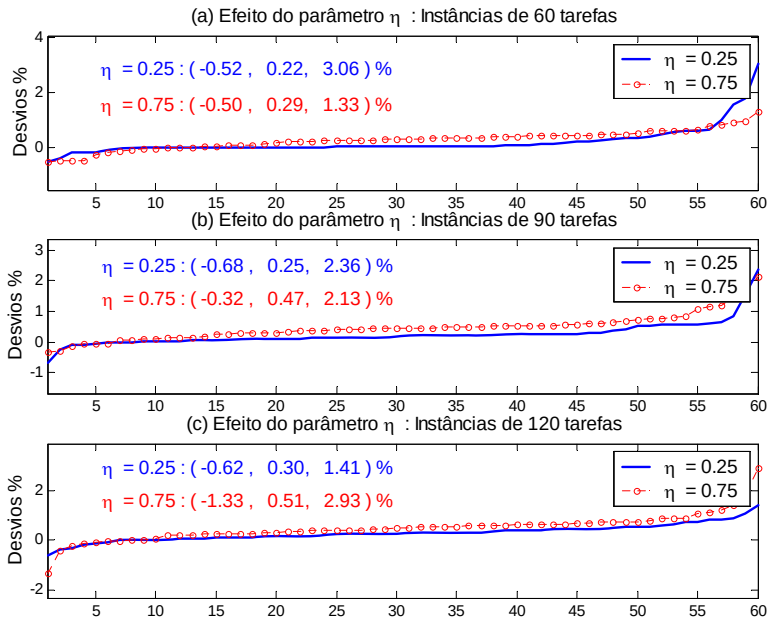

Figura 5.3 – PSET: GB *versus* GM – Efeito do parâmetro η

5.4 GRASP (GM) *versus* GRASP (GMP)

A realização de religações de caminhos, como estratégia de pós-otimização possibilitou a melhoria de 106 instâncias de 60 tarefas (58 com τ = 0.3 e 48 com τ = 0.9). Todas as instâncias de 90 e de 120 tarefas foram melhoradas.

Os desvios percentuais, $[Z(GM) - Z(GMP)]/Z(\min)$, são apresentados nas Figuras 5.4 e 5.5 segundo os parâmetros τ e R, respectivamente. Valores positivos são favoráveis à variante GMP. Como observado para o problema **PST**, a realização de religações de caminhos proporcionou desvios médios apenas marginais nas instâncias com τ = 0.9 (mais apertadas). Para tais instâncias, os maiores desvios são de 0.28% (instâncias de 60 tarefas), 0.83% (90 tarefas) e 0.68% (120 tarefas), enquanto que para as instâncias mais folgadas os maiores desvios chegaram a 1.93% (60 tarefas), 4.02% (90 tarefas) e 5.20% (120 tarefas) – Figura 5.4.

Em relação ao espalhamento das datas de entrega, Figura 5.5, os maiores desvios estão associados às instâncias com R = 0.75 (1.93% para as instâncias de 60 tarefas, 4.02% para as de 90 tarefas e 5.20% para as de 120 tarefas). Comportamento semelhante foi observado no item 5.3, na comparação entre GB e GM.

Foi observado, também, que o grau de severidade dos tempos de processamento é o que exerce menor influência no desempenho de GMP em comparação ao de GM.

Nas instâncias de 60 tarefas, a variante GMP gastou cerca de 94s (6% a mais que a variante GM). Nas instâncias de 90 tarefas, o tempo médio foi de 440s (acréscimo de 13%), enquanto que nas instâncias de 120 tarefas, o tempo médio foi de 1417s (aumento de 27%).

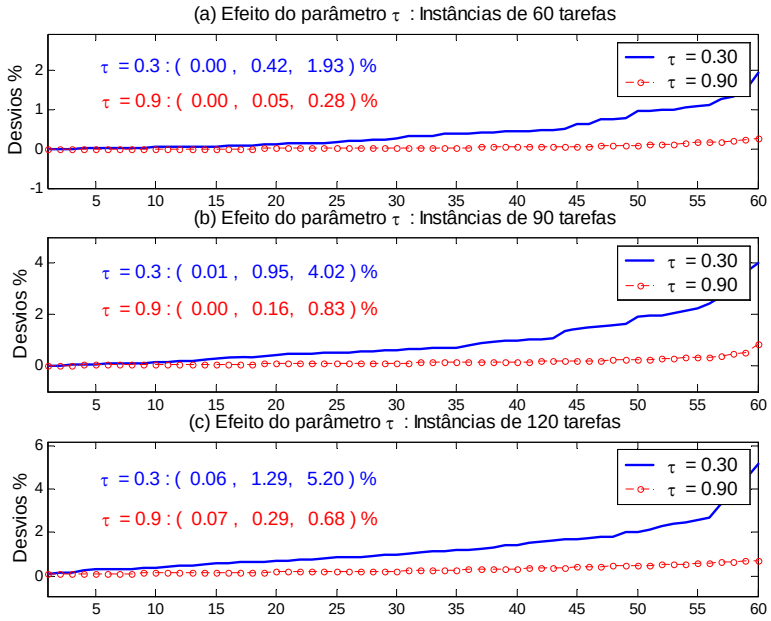

Figura 5.4 – PSET: GM *versus* GMP – Efeito do parâmetro τ

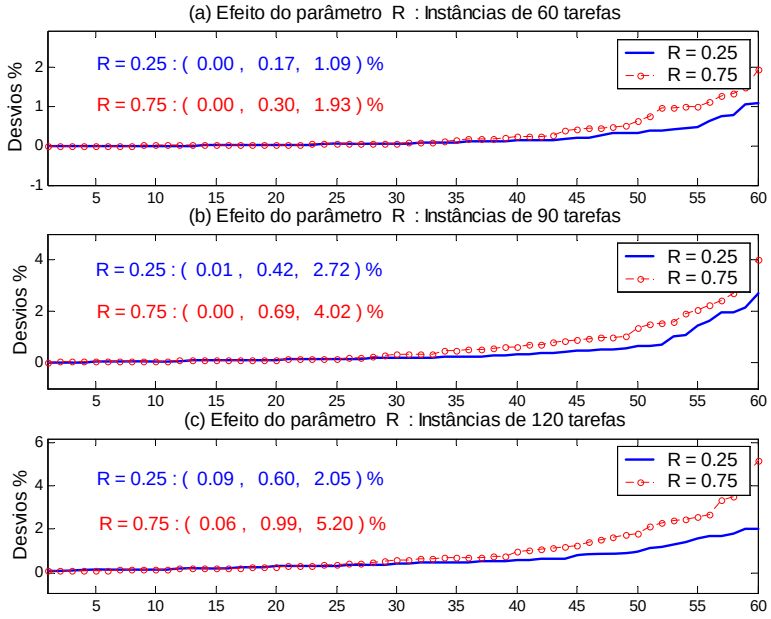

Figura 5.5 – PSET: GM *versus* GMP – Efeito do parâmetro R

5.5 GRASP (GM) *versus* GRASP (GMPop)

A aplicação de estratégia POP nas iterações da fase de aprendizado também possibilitou alguns melhores resultados em comparação a GM, mas não tanto quanto a aplicação de religações de caminho (pós-otimização). Dentre as 120 instâncias de 60 tarefas, GMPop promoveu a melhoria de resultados em 79. Os resultados piores totalizaram 34. No conjunto de instâncias de 90 tarefas, foram 82 vitórias de GMPop e 38 derrotas. Nas de 120 tarefas, GMPop venceu em 87 e perdeu em 33. Assim como ocorreu no problema **PST**, as melhorias proporcionadas por GMPop são menos impactantes que aquelas decorrentes do uso religações de caminho (pós-otimização).

Nas Figuras 5.6 e 5.7 mostram-se os desvios percentuais, para cada subconjunto de instâncias, em função dos parâmetros τ e R. Nestas figuras os desvios são calculados como $\left[Z(GM) - Z(GMPop)\right] / Z(\min)$. Valores positivos indicam vitórias de GMPop, e negativos indicam vitórias de GM. Em instâncias com τ = 0.9 os desvios são bem pequenos (máximo de 0.47% nas instâncias de 60 tarefas, 0.39% nas de 90 tarefas e 0.30% nas de 120 tarefas).

O teste de Wilcoxon foi aplicado a diversos subconjuntos de instâncias, comparando GM a GMPop. As conclusões estão apresentadas na Tabela 5.1, em que é indicada a variante dominante, para cada subconjunto de instâncias. Asteriscos indicam os subconjuntos nos quais não há dominância de uma variante sobre a outra.

Os tempos de CPU gastos na variante GMPop foram de 135s (60 tarefas), 650s (90 tarefas) e 1914s (120 tarefas). Estes resultados representam aumentos de 52%, 67% e 72%, em relação aos tempos gastos pela variante GM.

Tabela 5.1 – PSET: Teste de Wilcoxon (variantes dominantes) : GM *versus* GMPop

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	GMPop	GMPop	GMPop	GMPop	GMPop	GMPop
90	GMPop	GMPop	***	GMPop	GMPop	GMPop
120	GMPop	GMPop	GMPop	GMPop	GMPop	GMPop

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	***	GMPop	GMPop	***	GMPop
90	GMPop	GMPop	GMPop	GMPop	GMPop
120	GMPop	GMPop	GMPop	GMPop	GMPop

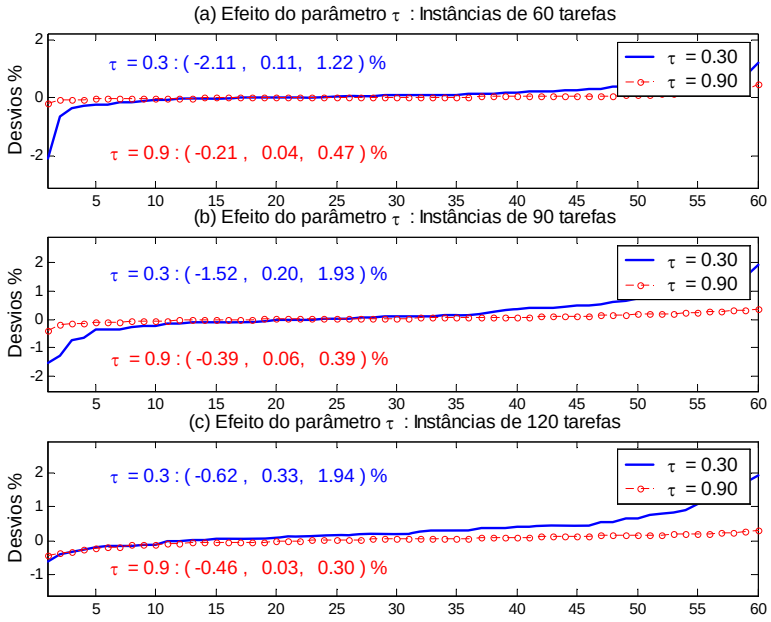

Figura 5.6 – PSET: GM *versus* GMPop – Efeito do parâmetro τ

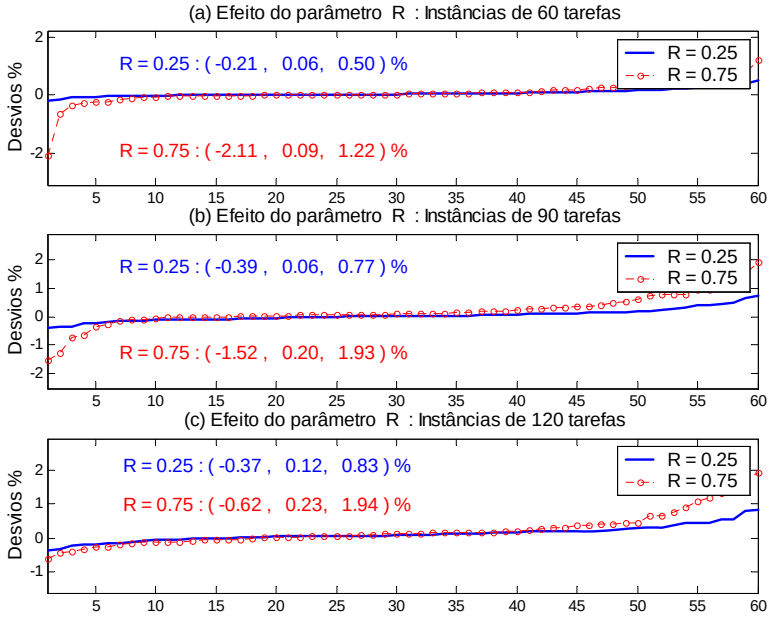

Figura 5.7 – PSET: GM *versus* GMPop – Efeito do parâmetro R

5.6 GRASP (GMP) *versus* GRASP (GMPP)

O desempenho da variante GMPP não foi muito superior ao da versão GMP. GMPP produziu melhores resultados em 64 instâncias de 60 tarefas, em 68 de 90 tarefas e em 61 de 120 tarefas. Por outro lado, as derrotas para a variante GMP ocorreram em 37 instâncias de 60 tarefas, em 52 instâncias de 90 tarefas e em 59 nas de 120 tarefas. Nas Figuras 5.8 e 5.9 mostram-se os desvios entre GMPP e GMP, para cada subconjunto de instâncias, definidos em função dos parâmetros τ e R. Os desvios são dados por $[Z(GMP) - Z(GMPP)]/Z(\min)$. Valores positivos indicam melhores resultados de GMPP. Como exemplo, tem-se, na Figura 5.8(a), que os desvios variam de –0.86% (favorável a GMP) a 1.59% (favorável a GMPP) com média de 0.08%, nas instâncias de 60 tarefas (τ = 0.3).

O teste de Wilcoxon, Tabela 5.2, indica a inexistência de dominância de uma variante sobre a outra, na maioria dos subconjuntos de instâncias.

Os tempos de CPU gastos na variante GMPP foram de 143s (60 tarefas), 688s (90 tarefas) e 2296s (120 tarefas). Estes resultados representam aumentos de 52%, 56% e 62%, em relação aos tempos gastos pela variante GMP.

Tabela 5.2 – PSET: Teste de Wilcoxon (variantes dominantes) : GMP *versus* GMPP						
Tarefas	$\tau = 0.3$	$\tau = 0.9$	R = 0.25	R = 0.75	$\eta = 0.25$	$\eta = 0.75$
60	***	GMPP	***	***	***	***
90	***	GMPP	***	GMPP	***	***
120	***	***	***	***	***	***
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	***	***	***	***	***	
90	***	***	***	***	***	
120	***	***	***	***	***	

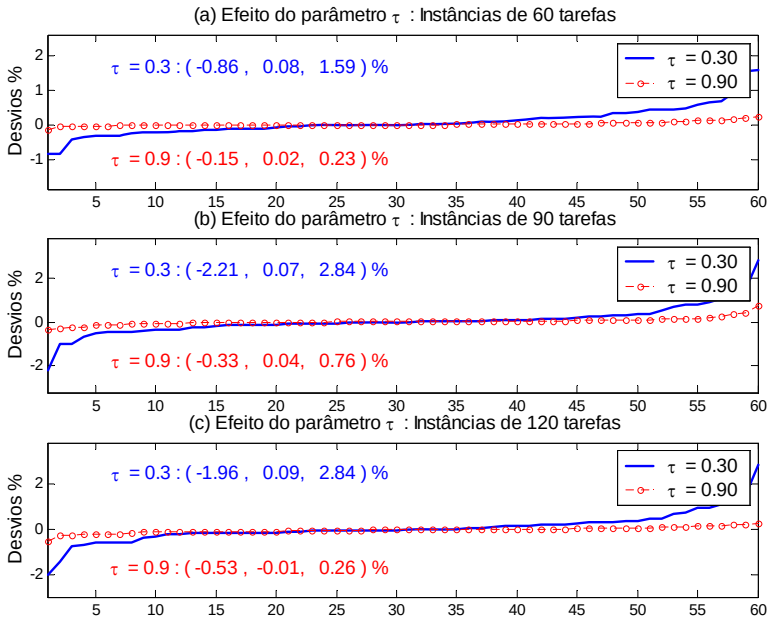

Figura 5.8 – PSET: GMP *versus* GMPP – Efeito do parâmetro τ

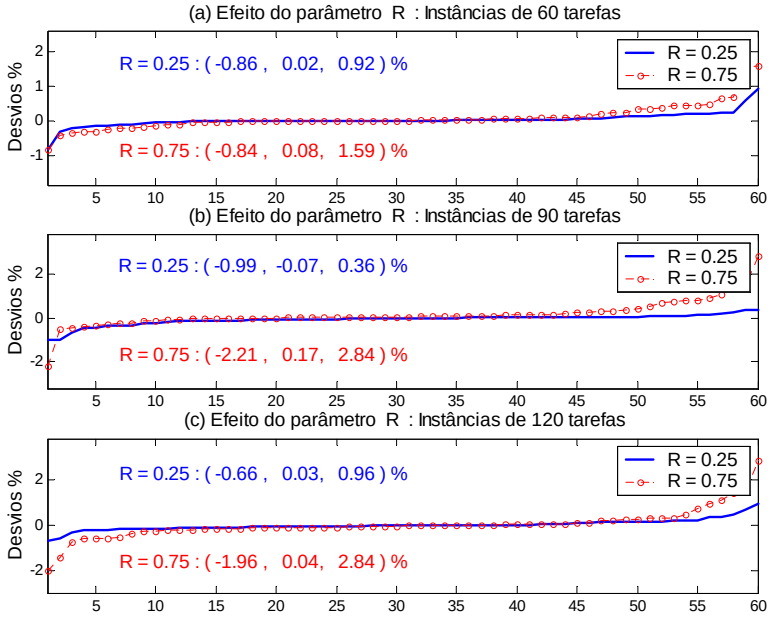

Figura 5.9 – PSET: GMP *versus* GMPP – Efeito do parâmetro R

5.7 GRASP (GB) *versus* Busca Tabu (TC)

O desempenho da Busca Tabu de curto prazo (TC) foi muito superior àquele do GRASP Básico (GB). TC vence GB em 99 instâncias de 60 tarefas e perde em 21 (destas últimas, 12 são instâncias com 12 máquinas). Nas instâncias de 90 tarefas, o número de vitórias de TC sobe para 108 e chega a 118 nas instâncias de 120 tarefas. Os desvios percentuais, $[Z(GB) - Z(TC)]/Z(\min)$, estão mostrados nas Figuras 5.10 e 5.11, para cada um dos parâmetros τ e R. Valores positivos são favoráveis a TC e negativos são favoráveis a GB. Na Figura 5.10 observa-se, por exemplo, que os desvios máximos (todos favoráveis a TC) foram de 0.57% para as instâncias de 60 tarefas com τ = 0.9, 0.94% para as de 90 tarefas e de 1.50% para as de 120 tarefas. As conclusões do teste de Wilcoxon são apresentadas na Tabela 5.3, confirmando a superioridade de TC sobre GB.

A exemplo do ocorrido no problema **PST**, a duração tabu calibrada para instâncias com 60 tarefas, 3 e 6 máquinas, foi excessiva para as instâncias de 60 tarefas e 12 máquinas, resultando no pior desempenho de TC nestas instâncias. Testes adicionais, aplicados a este

último conjunto de instâncias, com durações tabu menores (no intervalo $[0.0, 0.25n/m]$, para movimentos entre máquinas e no intervalo $[0.0, 0.05n*m]$, para movimentos internos) possibilitaram uma melhora significativa da Busca Tabu – foram 20 vitórias de TC contra apenas 4 de GB. O teste de Wilcoxon passa a indicar a dominância de TC sobre GB.

Enquanto os tempos médios gastos pelo GRASP Básico foram de 83s para instâncias com 60 tarefas, 360s para as de 90 tarefas e de 1015s para as de 120 tarefas, a Busca Tabu de curto prazo gastou, em média, 76s, 294s e 791s, permitindo economias de tempo computacional de 8%, 18% e 22%, respectivamente.

Tabela 5.3 – PSET: Teste de Wilcoxon (variantes dominantes) : GB *versus* TC

Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TC	TC	TC	TC	TC	TC
90	TC	TC	TC	TC	TC	TC
120	TC	TC	TC	TC	TC	TC

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TC	TC	TC	TC	*** (1) / TC (2)
90	TC	TC	TC	TC	TC
120	TC	TC	TC	TC	TC

(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6
(2) Com intervalos de duração tabu reduzidos

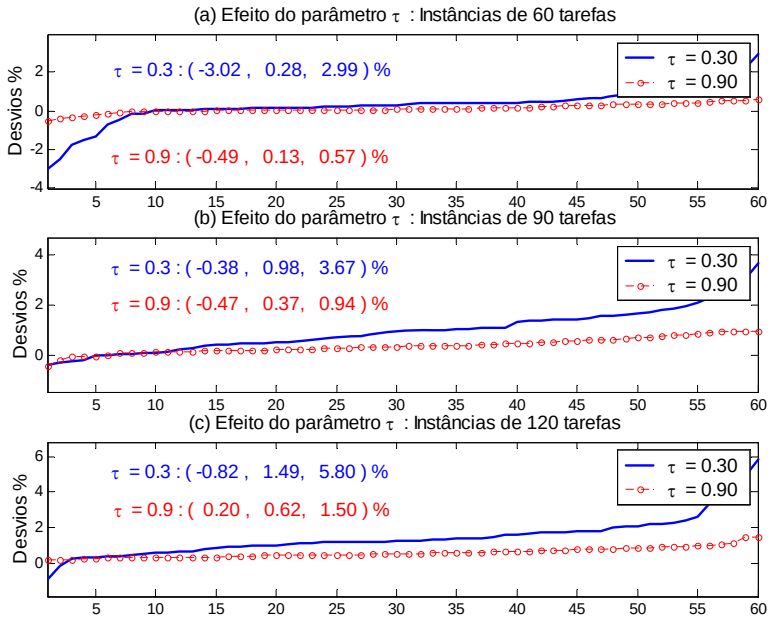

Figura 5.10 – PSET: GB *versus* TC – Efeito do parâmetro τ

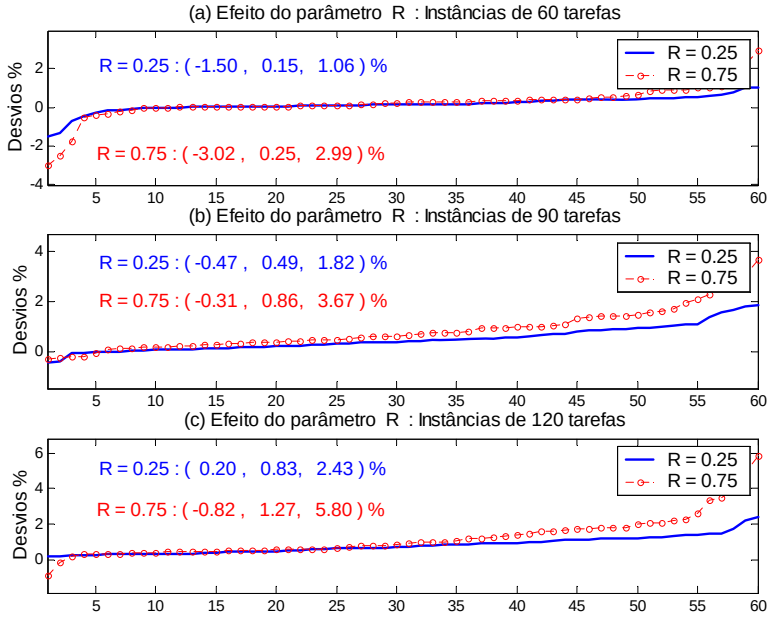

Figura 5.11 – PSET: GB *versus* TC – Efeito do parâmetro R

5.8 Busca Tabu (TC) *versus* Busca Tabu (TRT)

A variante TRT melhorou os resultados da Busca Tabu de curto prazo (TC) em cerca de 2/3 do total de instâncias. No terço restante os melhores resultados foram aqueles obtidos com TC (ou seja, quase não houve empates). Nas Figuras 5.12 e 5.13 os desvios percentuais entre TRT e TC, calculados como $\left[Z(TC) - Z(TRT)\right] / Z(\min)$, são apresentados. Valores positivos são favoráveis a TRT. O teste de Wilcoxon, cujas conclusões são mostradas na Tabela 5.4, indica a superioridade de TRT apesar do significativo número de derrotas.

Numa comparação com o problema **PST**, o desempenho menos expressivo de TRT pode ser atribuído ao fato das máquinas serem não relacionadas. O conjunto $\boldsymbol{T}$ de seqüências é mais rapidamente esvaziado e a solução parcial obtida com as seqüências do conjunto $\boldsymbol{T}$ é muito incompleta, ficando a cargo da regra de despacho uma parcela significativa do trabalho de factibilização da solução. Ou seja, as informações contidas nas boas seqüências do conjunto $\boldsymbol{T}$ são pouco exploradas. Vale a pena ressaltar que em Rochat e Taillard (1995) o procedimento é aplicado apenas a instâncias com frota homogênea.

A variante TRT gastou, em média, cerca de 11% mais tempo que a Busca Tabu de curto prazo. Os tempos médios ficaram em : 84s (instâncias de 60 tarefas), 327s (90 tarefas) e 877s (120 tarefas).

Tabela 5.4 – PSET: Teste de Wilcoxon (variantes dominantes) : TC *versus* TRT						
Tarefas	$\tau = 0.3$	$\tau = 0.9$	R = 0.25	R = 0.75	$\eta = 0.25$	$\eta = 0.75$
60	TRT	TRT	TRT	TRT	TRT	TRT
90	TRT	TRT	TRT	TRT	TRT	TRT
120	***	TRT	TRT	***	TRT	***
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	***	***	TRT	TRT	TRT	
90	***	TRT	TRT	***	TRT	
120	TRT	TRT	***	TRT	***	

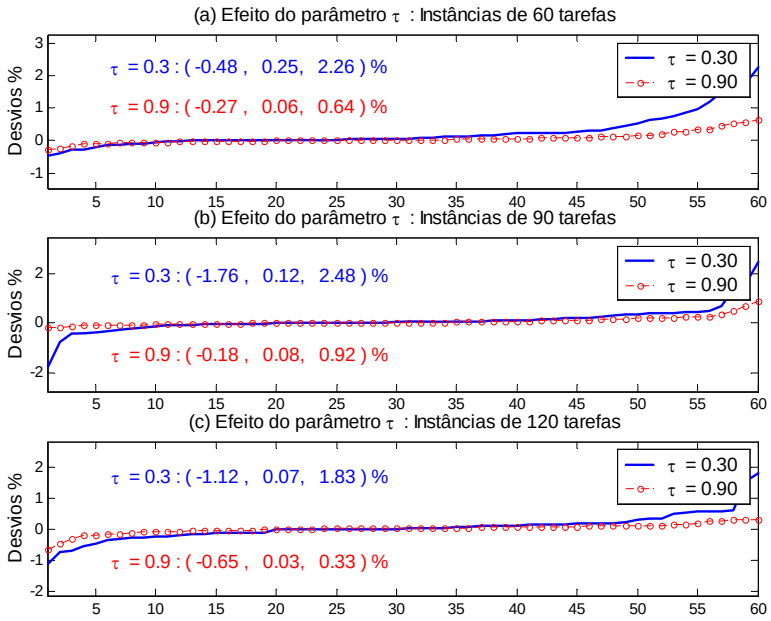

Figura 5.12 – PSET: TC *versus* TRT – Efeito do parâmetro τ

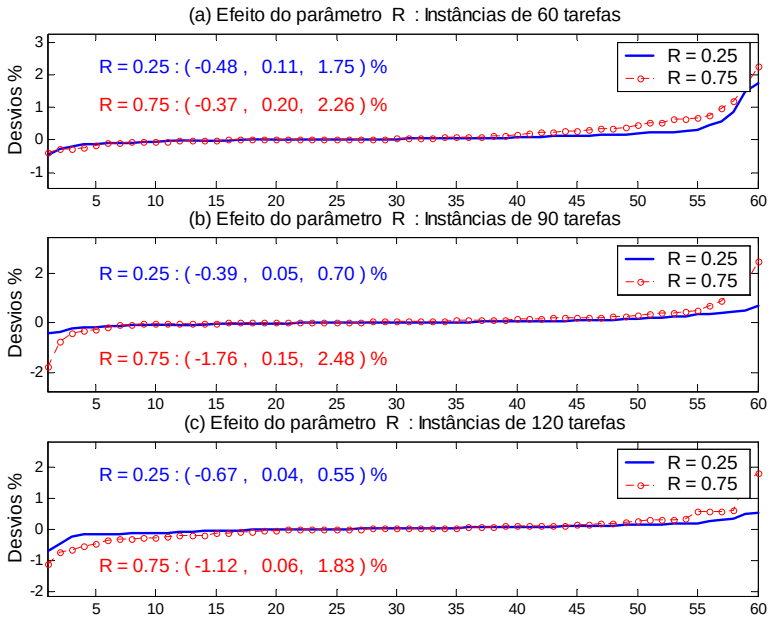

Figura 5.13 – PSET: TC *versus* TRT – Efeito do parâmetro R

5.9 Busca Tabu (TC) *versus* Busca Tabu (TDiv)

Foram realizados diversos testes, variando os parâmetros de controle da variante TDiv (frequência de ativação da fase de diversificação, duração da fase de diversificação, grau de penalização). De um modo geral, os resultados de TDiv foram ruins (piores que a Busca Tabu de curto prazo), confirmando a necessidade de uma estratégia de intensificação.

5.10 Busca Tabu (TC) *versus* Busca Tabu (TDivRC)

Os resultados de TDiv foram melhorados com a realização de religações intermediárias, após a obtenção de um mínimo local ao final da fase de diversificação. Ocorreram vitórias de TDivRC em 91 instâncias de 60 tarefas, em 93 instâncias de 90 tarefas e em 54 instâncias de 120 tarefas (neste último conjunto houve 45 empates). A superioridade de TDivRC sobre TC pode ser vista nas Figuras 5.14 e 5.15, que mostram os desvios percentuais, dados por $\left[Z(TC) - Z(TDivRC)\right] / Z(\min)$, bem como na Tabela 5.5, em que são mostradas as conclusões do teste de Wilcoxon. Desvios positivos são favoráveis a TDivRC.

A variante TDivRC gastou, em média, cerca de 13% mais tempo que a Busca Tabu de curto prazo, para as instâncias de 60 e 90 tarefas e cerca de 15% a mais para as instâncias de 120 tarefas. Os tempos médios ficaram em: 86s (instâncias de 60 tarefas), 338s (90 tarefas) e 913s (120 tarefas).

Tabela 5.5 – PSET: Teste de Wilcoxon (variantes dominantes) : TC *versus* TDivRC						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC
90	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC
120	TDivRC	***	TDivRC	TDivRC	TDivRC	***

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TDivRC	TDivRC	TDivRC	TDivRC	TDivRC
90	***	TDivRC	TDivRC	TDivRC	TDivRC
120	***	***	***	TDivRC	TDivRC

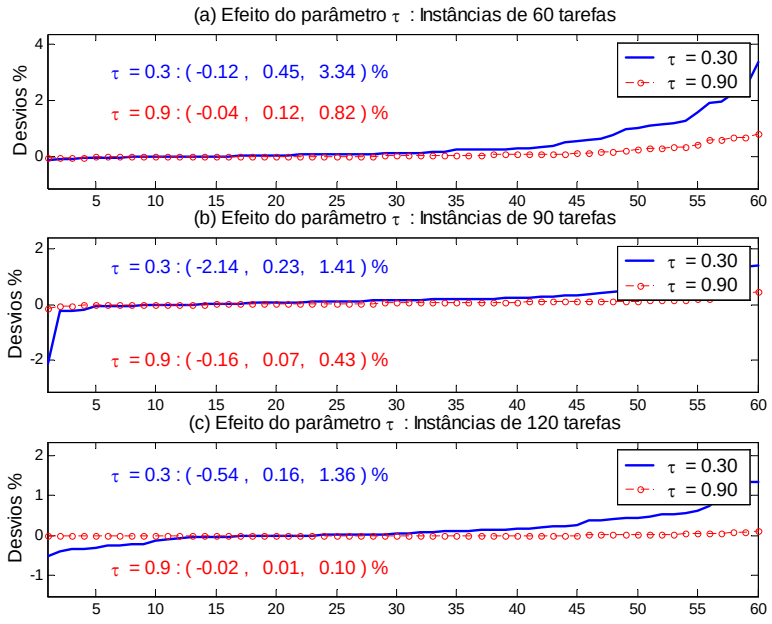

Figura 5.14 – PSET: TC *versus* TDivRC – Efeito do parâmetro τ

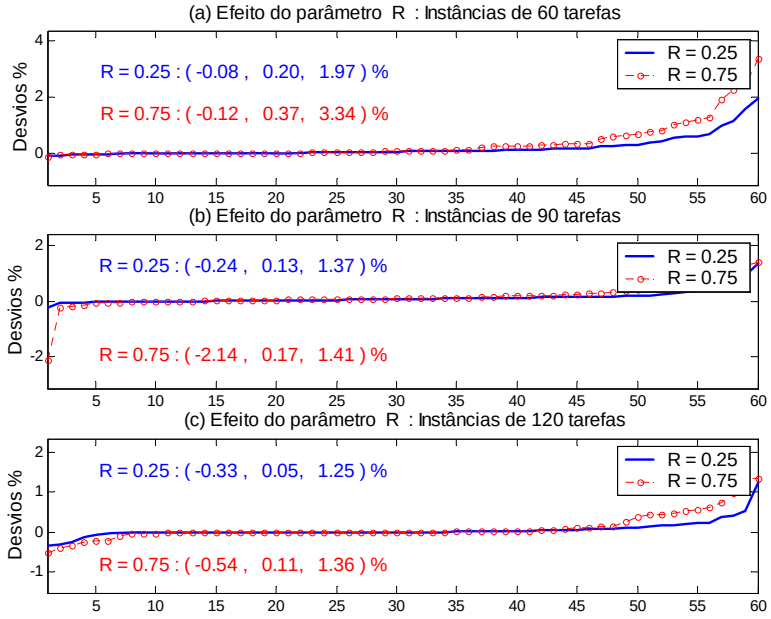

Figura 5.15 – PSET: TC *versus* TDivRC – Efeito do parâmetro R

5.11 Busca Tabu (TC) versus Busca Tabu (TRC)

A execução de religações de caminhos intermediárias, ao se detectar estagnação da busca, sem qualquer fase de diversificação, conferiu à variante TRC um desempenho muito superior. Ocorreram vitórias de TRC em 82 instâncias de 60 tarefas, em 94 instâncias de 90 tarefas e em 99 instâncias de 120 tarefas. TC foi melhor em apenas 7 instâncias de 60 tarefas, 19 de 90 tarefas e 18 de 120 tarefas. Os desvios percentuais, dados por $[Z(TC) - Z(TRC)]/Z(\min)$, alcançaram valores de até 8.07% (instâncias de 60 tarefas), 7.39% (instâncias de 90 tarefas) e 2.87% (instâncias de 120 tarefas). O teste de Wilcoxon aponta dominância de TRC em todos os subconjuntos de instâncias considerados.

5.12 Busca Tabu (TDivRC) *versus* Busca Tabu (TRC)

A Busca Tabu acrescida apenas das religações de caminhos intermediárias (TRC) mostrou-se melhor que a variante com diversificação e religações de caminhos (TDivRC). TRC foi melhor em 78 instâncias de 60 tarefas, em 93 de 90 tarefas e em 100 de 120 tarefas, tendo sido superada em 14 instâncias de 60 tarefas, 26 de 90 tarefas e 19 de 120 tarefas. Os desvios percentuais, $\left[Z(TDivRC) - Z(TRC)\right]/Z(\min)$, favoráveis à variante TRC, quando positivos, foram de 0.28% (média) e 5.44% (máximo) nas instâncias de 60 tarefas, 0.30% (média) e 4.31% (máximo) nas instâncias de 90 tarefas e 0.29% (média) e 3.07% (máximo) nas instâncias de 120 tarefas. O teste de Wilcoxon aponta dominância de TRC em todos os subconjuntos de instâncias considerados.

No item 5.9 foi observado que o efeito isolado da diversificação baseada em freqüência de residência não surtiu o efeito esperado, ou seja, regiões contendo soluções de boa qualidade e estruturalmente bem diferentes da região explorada pela estratégia de curto prazo, não foram alcançadas após a fase de diversificação ter sido desativada. Conseqüentemente, houve uma deterioração na qualidade do conjunto de elite que teve reflexos também negativos na fase de religação de caminhos e que culminou no desempenho inferior de TDivRC, em comparação ao de TRC.

5.13 Busca Tabu (TRT) *versus* Busca Tabu (TRC)

A Busca Tabu com religações intermediárias (TRC) também apresentou desempenho superior frente a variante de Busca Tabu com reinícios segundo Rochat e Taillard (TRT). Tais resultados são apresentados nas Figuras 5.16 e 5.17, na forma de desvios percentuais, $\left[Z(TRT) - Z(TRC)\right]/Z(\min)$, bem como na Tabela 5.6 (teste de Wilcoxon). Nas Figuras 5.16 e 5.17, valores positivos são favoráveis à variante TRC. A variante TRC obteve melhores resultados em 92 instâncias de 60 tarefas, em 91 de 90 tarefas e em 92 de 120 tarefas, tendo sido superada em apenas 9 instâncias de 60 tarefas, em 28 de 90 tarefas e em 28 de 120 tarefas.

Na comparação entre os tempos de CPU das variantes TRT e TRC, esta última gastou, em média, 2% mais tempo que a primeira.

Tabela 5.6 – PSET: Teste de Wilcoxon (variantes dominantes) : TRT *versus* TRC						
Tarefas	$\tau = 0.3$	$\tau = 0.9$	R = 0.25	R = 0.75	$\eta = 0.25$	$\eta = 0.75$
60	TRC	TRC	TRC	TRC	TRC	TRC
90	TRC	TRC	TRC	TRC	TRC	TRC
120	TRC	TRC	TRC	TRC	TRC	TRC
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	TRC	TRC	TRC	TRC	TRC	
90	***	TRC	TRC	TRC	TRC	
120	TRC	TRC	TRC	TRC	TRC	

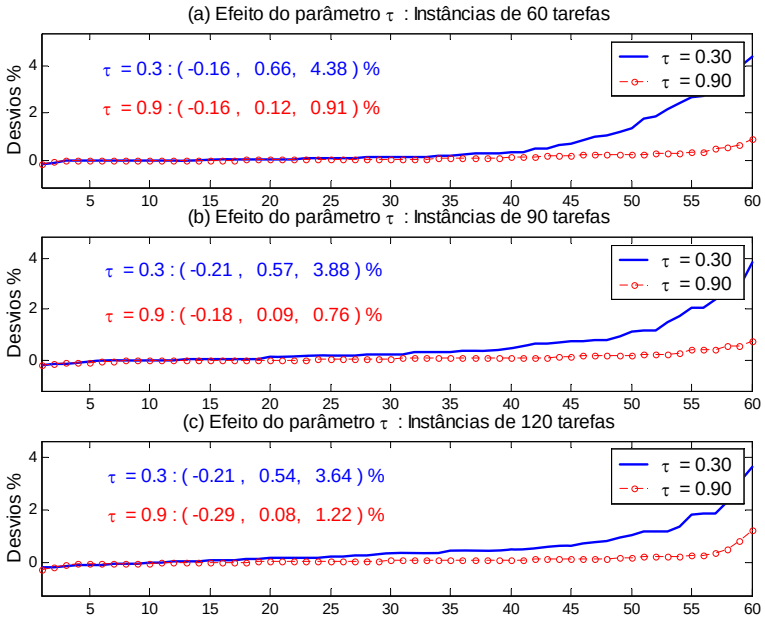

Figura 5.16 – PSET: TRT *versus* TRC – Efeito do parâmetro τ

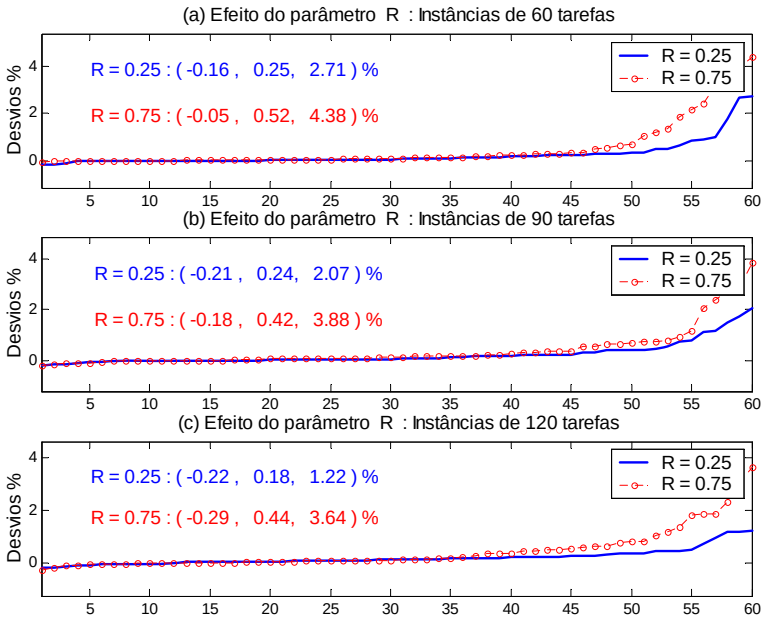

Figura 5.17 – PSET: TRT *versus* TRC – Efeito do parâmetro R

5.14 GRASP (GMP) *versus* Busca Tabu (TRTP)

Comparando-se o GRASP com memória acrescido de religações de caminhos na pós-otimização (GMP) e a Busca Tabu de longo prazo (TRT) também acrescida de religações de caminhos, observa-se uma vantagem em favor da Busca Tabu nas instâncias com τ = 0,3 (144 vitórias da Busca Tabu contra 33 vitórias do GRASP e 6 empates) e um equilíbrio nas instâncias com τ = 0,9 (90 vitórias da Busca Tabu contra 82 vitórias do GRASP e 8 empates). Os desvios percentuais, $\left[Z(GMP) - Z(TRTP)\right] / Z(\min)$, são mostrados nas Figuras 5.18 e 5.19, com valores positivos favoráveis à Busca Tabu. As conclusões do teste de Wilcoxon são apresentadas na Tabela 5.7, em que se observa, dentre outros, que não há dominância de uma meta-heurística sobre a outra nas instâncias com τ = 0,9, nem naquelas com 12 máquinas. Deve ser observado que o desempenho da Busca Tabu foi prejudicado em particular nas instâncias com 12 máquinas, em decorrência da duração tabu excessiva e, de um modo geral, pelo fato das máquinas não serem idênticas (como no item 5.8). Testes adicionais com durações tabu reduzidas (mesmos intervalos adotados no item 5.7) indicam uma pequena melhora da Busca Tabu, mas não a ponto de dominar o GMP.

Os tempos de GMP foram, em média, 94s, 440s e 1418s para as instâncias de 60, 90 e 120 tarefas, enquanto que os tempos de TRTP ficaram em 87s, 351s e 993s.

Tabela 5.7 – PSET: Teste de Wilcoxon (variantes dominantes) : GMP *versus* TRTP						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TRTP	***	***	TRTP	TRTP	TRTP
90	TRTP	***	***	TRTP	***	TRTP
120	TRTP	***	TRTP	TRTP	TRTP	TRTP

Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12
60	TRTP	TRTP	TRTP	TRTP	*** (1) / *** (2)
90	***	TRTP	TRTP	TRTP	*** (1) / *** (2)
120	TRTP	TRTP	TRTP	***	*** (1) / *** (2)

(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6
(2) Com intervalos de duração tabu reduzidos

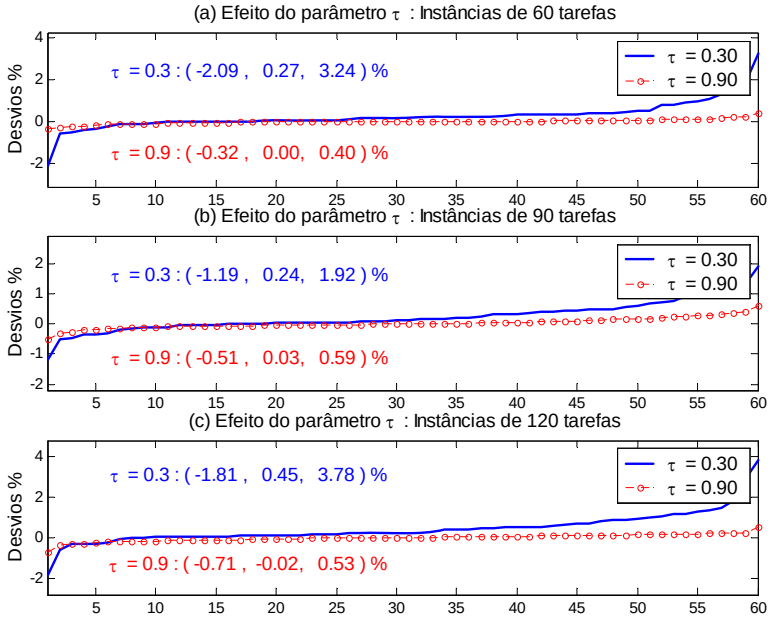

Figura 5.18 – PSET: GMP *versus* TRTP – Efeito do parâmetro τ

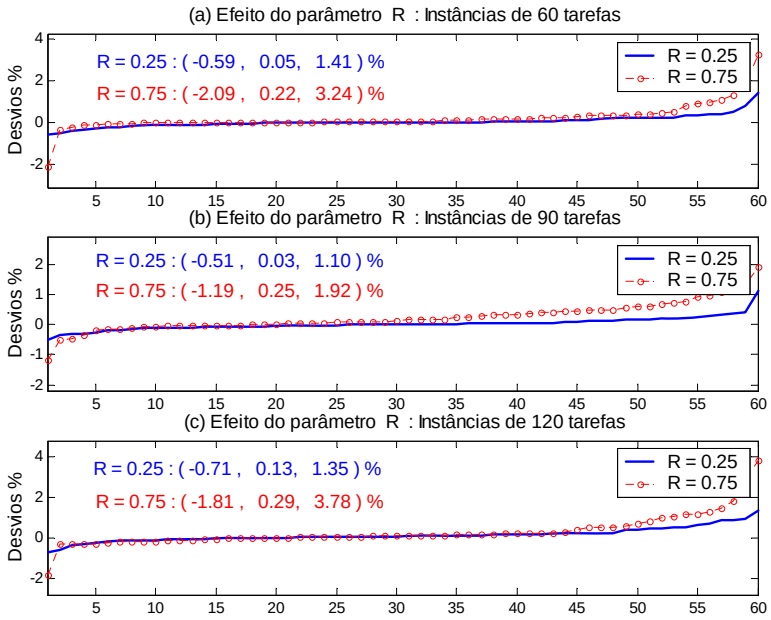

Figura 5.19 – PSET: GMP *versus* TRTP – Efeito do parâmetro R

5.15 GRASP (GMP) versus Busca Tabu (TRCP)

O desempenho da Busca Tabu de longo prazo (TRCP) foi melhor que o do GRASP (GMP). Destaque para as instâncias com τ = 0.3, nas quais houve 157 vitórias da Busca Tabu, 20 vitórias do GRASP e 3 empates. Para as instâncias com τ = 0.9 foram 122 vitórias da Busca Tabu contra 46 do GRASP e 12 empates. Os desvios percentuais entre GMP e TRCP estão mostrados nas Figuras 5.20 e 5.21. Valores positivos são favoráveis à variante de Busca Tabu. As conclusões do teste de Wilcoxon são mostradas na Tabela 5.8. O desempenho de TRCP, nas instâncias de 12 máquinas, é melhorado quando durações tabu menores (mesmos intervalos adotados no item 5.7) são empregados.

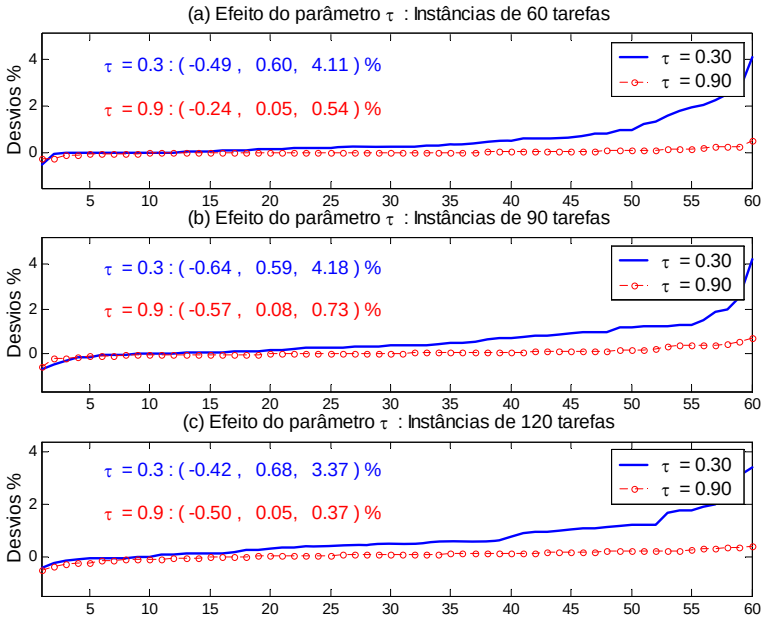

Figura 5.20 – PSET: GMP *versus* TRCP – Efeito do parâmetro τ

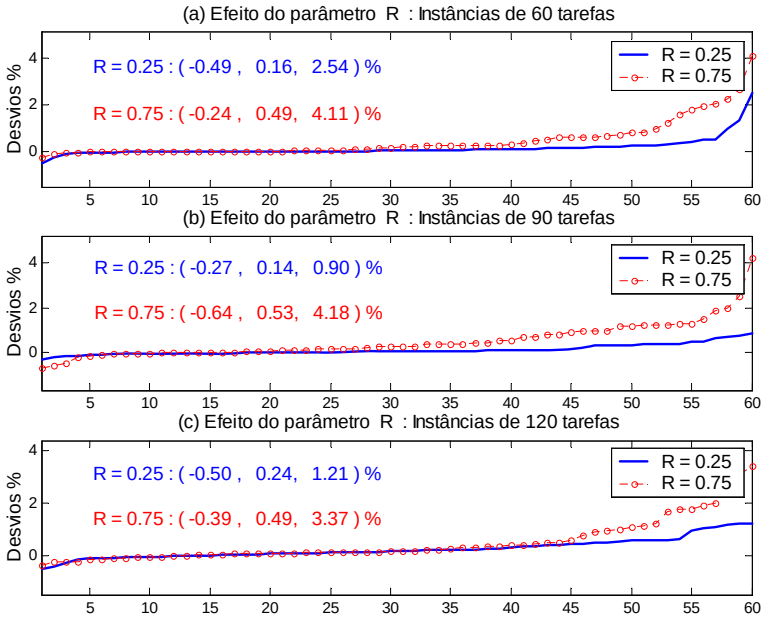

Figura 5.21 – PSET: GMP *versus* TRCP – Efeito do parâmetro R

Tabela 5.8 – PSET: Teste de Wilcoxon (variantes dominantes) : GMP *versus* TRCP						
Tarefas	τ = 0.3	τ = 0.9	R = 0.25	R = 0.75	η = 0.25	η = 0.75
60	TRCP	TRCP	TRCP	TRCP	TRCP	TRCP
90	TRCP	TRCP	TRCP	TRCP	TRCP	TRCP
120	TRCP	TRCP	TRCP	TRCP	TRCP	TRCP
Tarefas	m = 2	m = 3	m = 4	m = 6	m = 12	
60	TRCP	TRCP	TRCP	TRCP	TRCP	
90	TRCP	TRCP	TRCP	TRCP	TRCP	
120	TRCP	TRCP	TRCP	TRCP	*** (1) / TRCP (2)	
(1) Com intervalos de duração tabu calibrados para as instâncias 60x3 e 60x6 (2) Com intervalos de duração tabu reduzidos						

5.16 Médias dos tempos de CPU – Resumo

As médias dos tempos de CPU (em segundos) das diversas variantes implementadas são apresentadas nas Figuras 5.22, 5.23 e 5.24, para as instâncias de 60, 90 e 120 tarefas, respectivamente. As estratégias implementadas objetivando melhorias nos desempenhos das versões básicas, tanto do GRASP, quanto da Busca Tabu, impõem acréscimos nas médias dos tempos de CPU, mas estes são compensados pelos desempenhos superiores. A única exceção está associada à estratégia POP, cuja relação custo-benefício não se mostrou satisfatória.

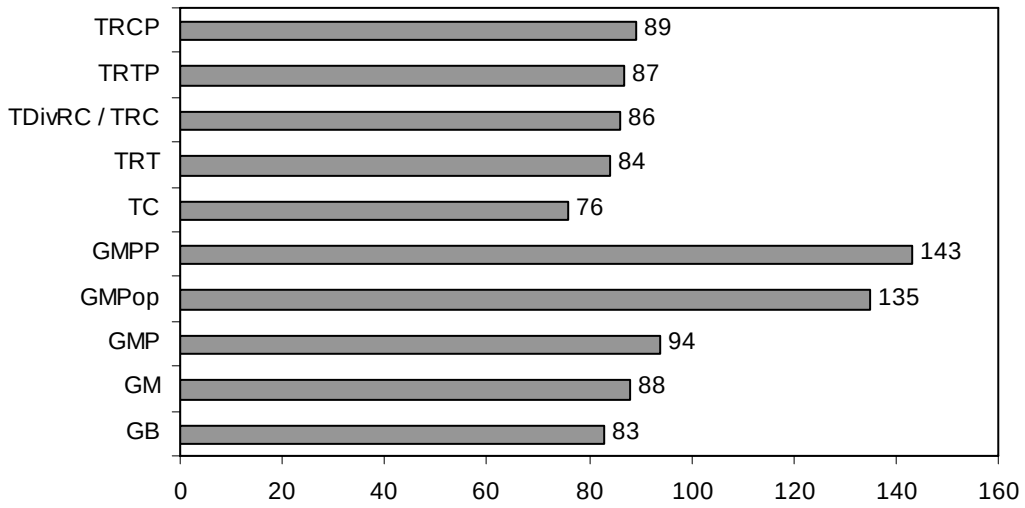

Figura 5.22 – PSET: Médias dos tempos de CPU (s) para as instâncias de 60 tarefas

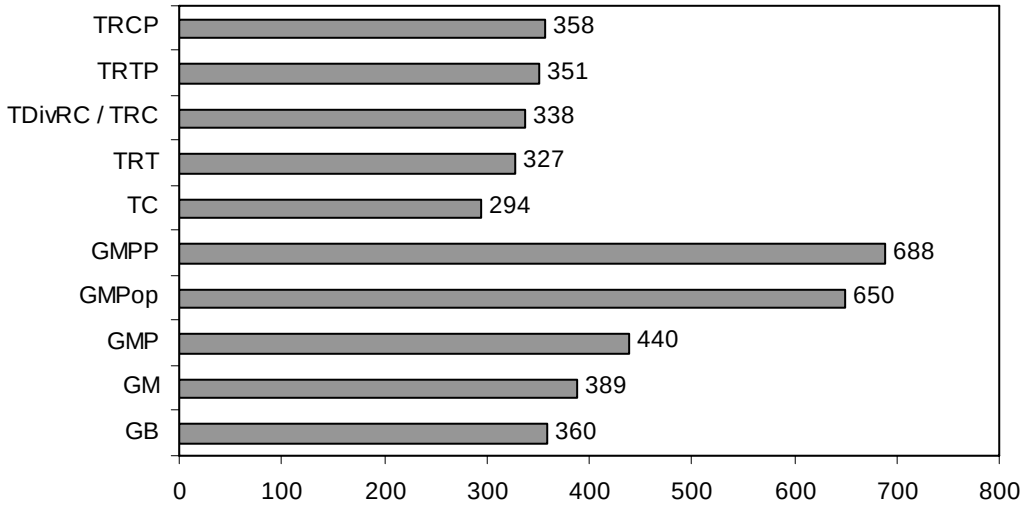

Figura 5.23 – PSET: Médias dos tempos de CPU (s) para as instâncias de 90 tarefas

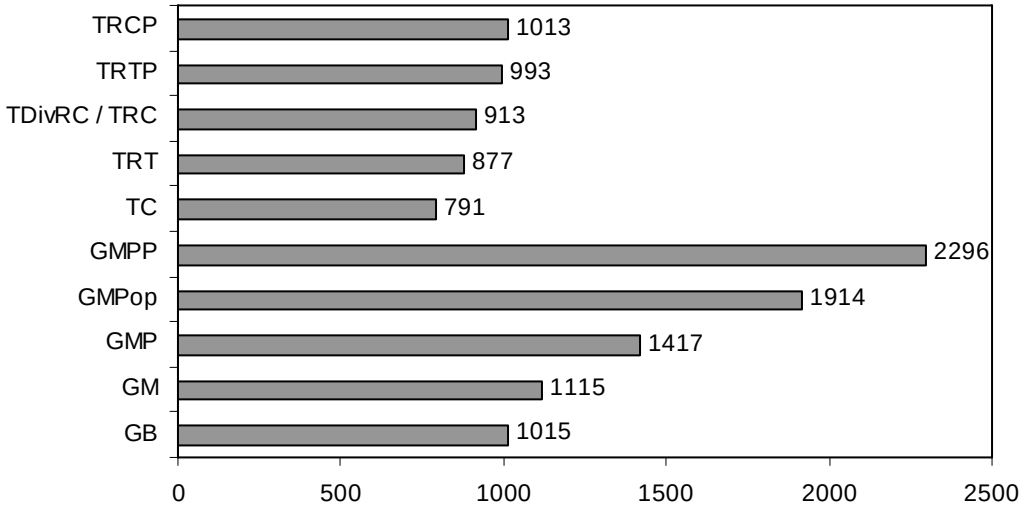

Figura 5.24 – PSET: Médias dos tempos de CPU (s) para as instâncias de 120 tarefas

Capítulo 6

Conclusões e trabalhos futuros

6.1 Conclusões

Neste trabalho foram estudados dois problemas clássicos de programação de tarefas em máquinas paralelas. O primeiro deles, denominado **PST** (das palavras-chave em inglês *Parallel machines*, *Setup times* e *Tardiness*) tem por objetivo a minimização de custos de atraso, enquanto que no segundo, denominado **PSET** (das palavras-chave em inglês *Parallel machines*, *Setup times, Earliness* e *Tardiness*), o objetivo é a minimização de custos de avanço e de atraso. Para resolvê-los, duas meta-heurísticas, GRASP e Busca Tabu, foram implementadas.

Em termos de GRASP foram avaliadas uma implementação básica e uma outra que utiliza um conjunto de soluções de elite para influenciar a fase construtiva. A ponderação entre a função gulosa e a função de intensidade ocorre de forma adaptativa, a partir da entropia associada às soluções obtidas ao longo de um certo conjunto de iterações. Também foi avaliado o princípio da otimalidade próxima, no caso presente implementado como buscas locais em soluções parciais. Como estratégia de pós-otimização foram empregadas religações de caminhos entre as soluções de elite.

Na Busca Tabu foram investigados dois critérios de proibição baseados em relações de precedência e um baseado nas atribuições das tarefas às máquinas. As estratégias de longo prazo incluíram diversificação por freqüência de residência, religações de caminhos durante a busca e um procedimento adaptado do problema de roteamento de veículos, em que novas soluções são criadas a partir de um banco de seqüências de tarefas, visando uma maior interação entre diversificação e intensificação. Também a Busca Tabu foi submetida a uma pós-otimização via religações de caminhos.

Nas buscas locais, empregadas nas duas meta-heurísticas, foram investigados movimentos comumente utilizados em problemas de programação de produção (trocas e inserções), bem como movimentos utilizados em problemas de roteamento de veículos (*cross* e *Or-Opt*). Para cada uma das meta-heurísticas foram feitos dois conjuntos de implementações. Um com movimentos de troca e de inserção, executados alternadamente, outro com movimentos *cross* e *Or-Opt*, também executados alternadamente. As vizinhanças geradas pelos movimentos *cross* e *Or-Opt* são bem maiores que as outras duas. Esperava-se, em princípio, que as implementações nelas baseadas apresentassem melhor desempenho, nos dois problemas tratados. Entretanto, os experimentos realizados mostraram que as implementações com movimentos *cross* e *Or-Opt* tiveram um desempenho pior no problema **PSET**. Em tal problema, até mesmo pequenas modificações estruturais em uma solução podem produzir grandes variações no custo. Portanto, movimentos menos impactantes são necessários para a exploração mais efetiva do espaço de soluções. Por outro lado, no problema **PST**, pequenas modificações estruturais promovem pequenas variações no custo. Logo, movimentos com maior poder de modificação estrutural da solução podem, e até devem, ser executados a fim de (mais rapidamente) levar a busca para regiões mais promissoras.

A alocação temporal das tarefas, no problema **PSET**, exige a inserção de tempos ociosos. Neste trabalho foi desenvolvido um procedimento cuja principal vantagem decorre do fato de ser executado em duas etapas. Na primeira, de complexidade polinomial, promove-se a minimização dos custos de atraso e a conseqüente maximização dos custos de avanço. Na segunda, inserem-se os tempos ociosos. A segunda etapa é executada de modo seletivo e pode ser abandonada em qualquer estágio, possibilitando uma redução do custo computacional.

Em relação às estratégias para melhorar o desempenho do GRASP básico, são as seguintes as conclusões permitidas por este trabalho:

- O uso de informações do conjunto de elite, via função de intensidade, para influenciar a construção das soluções de partida, foi altamente satisfatório. Contudo, tão importante quanto isto é a ponderação adequada entre a função gulosa e a função de intensidade. Nesse sentido, a utilização da função de entropia foi bem sucedida;

- O procedimento mais flexível para a construção do conjunto de elite, proposto neste trabalho, conduziu a resultados ligeiramente melhores que aqueles alcançados com o procedimento clássico;

- O princípio de otimalidade próxima (POP) não se mostrou tão atraente como esperado. Além das soluções finais não terem sido sistematicamente melhores que as soluções obtidas sem a utilização do POP, o esforço computacional cresceu de modo significativo;

- As religações de caminho foram muito eficientes na melhoria de qualidade das soluções, sem impor um aumento excessivo no esforço computacional.

Quanto à Busca Tabu, as principais observações dizem respeito às estratégias de longo prazo:

- O efeito isolado da diversificação baseada em freqüência de residência não surtiu por completo o efeito esperado, ou seja, regiões contendo soluções de boa qualidade e estruturalmente bem diferentes da região explorada pela estratégia de curto prazo, não foram alcançadas após a fase de diversificação ter sido desativada. Como conseqüência, houve uma deterioração na qualidade do conjunto de elite, cujos reflexos negativos sobre as religações de caminho culminaram no desempenho aquém do esperado, frente às demais estratégias de longo prazo;

- O procedimento adaptado do trabalho de Rochat e Taillard (1995), que consiste na realização de reinícios a partir de soluções criadas por meio de um banco de seqüências, não se mostrou robusto, embora seja promissor. Foi observado que o percentual de máquinas não preenchidas ao final da segunda fase do procedimento de Rochat e Taillard (fase de extração de seqüências) é maior nas instâncias mais apertadas em termos da média das datas de entrega ($\tau = 0.9$). Também foi observado que o percentual de reinícios nos quais era obtida uma solução completa, ao final da segunda fase, com seqüências extraídas de diferentes soluções de elite, é menor em tais instâncias. Ou seja, nas instâncias mais apertadas, o procedimento de extração de seqüências ou preenche poucas máquinas ou preenche todas elas, mas a partir de uma única solução de elite. Neste último caso, o procedimento perde sua capacidade de promover interação entre diversificação e intensificação. Estes dois fatores colaboraram para uma perda de eficiência do

procedimento. As análises mostraram, ainda, que quanto maior o total de máquinas, maior o percentual daquelas não preenchidas na fase de extração de seqüências e, portanto, maior a porção da solução que precisa ser completada pela fase de factibilização, alicerçada em uma simples regra de despacho.

- Ainda em relação ao procedimento adaptado de Rochat e Taillard, foi observado que seu desempenho em problemas com máquinas não relacionadas fica comprometido porque, independente da instância ser mais ou menos apertada, ter muitas ou poucas máquinas, o conjunto de seqüências é mais rapidamente esvaziado e a solução parcial obtida com as seqüências do conjunto é muito incompleta, ficando a cargo da regra de despacho uma parcela significativa do trabalho de factibilização da solução. Ou seja, as informações contidas nas boas seqüências do conjunto de elite são pouco exploradas.

- Os resultados da variante constituída apenas por movimentos guiados pelas regras de curto prazo acrescidos de religações de caminhos ao se detectar estagnações, foram ou de qualidade semelhante (problema **PST**) ou melhores (problema **PSET**) que aqueles obtidos com as duas outras variantes, ambas mais elaboradas. Tais resultados servem como alerta para não se deixar de lado as estratégias mais simples.

Na comparação entre a melhor implementação GRASP, com memória e pós-otimização via religações de caminhos, e quaisquer das implementações de Busca Tabu com estratégias de longo prazo, foi observado um melhor desempenho da Busca Tabu, tanto para o problema **PST**, quanto para o problema **PSET**.

A principal contribuição do presente trabalho reside no tratamento do problema **PSET**, que, até onde pôde ser apurado no levantamento bibliográfico, não é abordado, por meio de meta-heurísticas, em um único trabalho, com as mesmas e todas particularidades aqui consideradas. Como outras contribuições, ainda que pontuais, tem-se:

- A utilização, em problemas de programação de tarefas, de movimentos comumente utilizados em roteamento de veículos;

- O procedimento de alocação temporal das tarefas de uma dada seqüência, com inserção de tempos ociosos;

- O procedimento flexível para construção do conjunto de elite;
- A aplicação do procedimento proposto por Rochat e Taillard, em um contexto de Busca Tabu, a problemas de programação de tarefas, em especial um com máquinas não relacionadas – que guarda equivalência com o roteamento de veículos com frota heterogênea – muito embora tenham sido detectadas algumas limitações.

6.2. Trabalhos futuros

As propostas para trabalhos futuros constituem tentativas de superar algumas das limitações observadas na implementação de Busca Tabu com estratégia de longo prazo baseada no procedimento proposto por Rochat e Taillard:

- Utilização de memória, nos mesmos moldes empregados no GRASP, na fase de factibilização das soluções parciais;
- No problema **PSET**, que sofre pelo esvaziamento muito acentuado do banco de seqüências, pelo fato das máquinas serem não relacionadas, pode-se definir m bancos de seqüências, um para cada máquina, de modo que ao se escolher uma seqüência de um banco, apenas as demais seqüências do referido banco sejam eliminadas. As seqüências dos demais bancos têm o seu indicador, que é utilizado para estabelecer a probabilidade de sua escolha, reavaliado de alguma forma, de modo a diminuir a probabilidade de escolha daquelas seqüências que contém tarefas presentes nas seqüências já escolhidas. Assim, ao final da fase de extração de seqüências, em lugar de uma solução incompleta pode haver uma solução infactível pela redundância de tarefas. A factibilização pode ser feita retirando as tarefas redundantes, de acordo com o procedimento proposto por Heady e Zhu (1998).

Referências bibliográficas

Abdul-Razaq, T. S., Potts, C. N., Van Wassenhove, L. N. 1990. A survey of algorithms for the single machine total weighted tardiness problem. *Discrete App. Mathematics*, 26, pp. 235-253.

Ahmadi, S., Osman, I. H. 2005. Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research*, 162, pp. 30-44.

Allaverdi A., Gupta J. N. D., Aldowaisan T. 1999. A review of scheduling research involving setup considerations. *OMEGA*, 27, pp. 219-239.

Almeida, M. T., Centeno, M. 1998. A composite heuristic for the single machine early/tardy job scheduling problem. *Computers and Operations Research*, 25, pp.625-635.

Anghinolfi, D., Paolucci, M. (In press). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research*.

Anglani, A., Grieco, A., Guerriero, E., Musmanno, R. 2005. Robust scheduling of parallel machines with sequence-dependent set-up costs. *European Journal of Operational Research*, 161, pp. 704-720.

Armentano, V. A., França Filho, M. F. (2007). Minimizing Total Tardiness in Parallel Machine Scheduling with Setup Times: An Adaptive Memory- Based GRASP Approach. *European Journal of Operational Research*, 183, pp. 100-114.

Baker, K. R., Scudder, G. D. 1990. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 28, pp. 22-36.

Balakrishnan, N., Kanet, J. J., Sridharan, S. V. 1999. Early/tardy scheduling with sequence dependent setups on uniform parallel machines, *Computers and Operations Research*, 26, pp. 127-141.

Bank, J., Werner, F. 2001. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. *Mathematical and Computer Modelling*, 33 (4/5), pp. 363-383.

Bilge, Ü., Kiraç, F., Kurtulan, M., Pekgün, P. 2004. A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research*, 31, pp. 397-414.

Binato. S., Hery, J., Loewenstern, D. M., Resende, M. G. C. 2002. A greedy randomized adaptive search procedure for job shop scheduling. *In: Ribeiro CC, Hansen P (Eds), Essays and Surveys on Metaheuristics, Kluwer*, pp. 58-79.

Bräysy, O. 2003. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing* , 15 pp. 347-368.

Bräysy, O., Gendreau, M. 2005. Vehicle routing problems with time windows, Part II: Metaheuristics. *Transportation Science*, 39; pp. 119-139.

Bresina, J. L. 1996. Heuristic-biased stochastic sampling. *Proceedings of the AAAI-96*, 1, pp. 271-278.

Burke, E.K., Gustafson, S., Kendall, G. 2004. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8, pp. 47-61.

Carlyle, W. M., Kim, B., Fowler, J. W., Gel, E.S. 2001. Comparison of multiple objective genetic algorithms for parallel machine scheduling problems. *Lecture Notes in Computer Science*, 1993, pp. 472-485.

Chen, Z. L., Powell, W. B. 1999. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operational Research*, 116, pp. 220-232.

Cheng, T. C. E., Chen, Z. L., Shakhlevich, N. V. 2002. Common due date assignment and scheduling with ready times. *Computers and Operations Research*, 29/14, pp. 1957-1967.

Cheng, T. C. E., Sin, C. C. S. 1990. A sate-of-the-art review of parallel-machine scheduling research. *European Journal of Operations Research*, 47, pp. 271-292.

Christofoletti, L. M. 2002. Métodos de reinício aplicados ao problema de sequenciamento em uma máquina com tempos de preparação e datas de entrega. *Dissertação de mestrado. Universidade Estadual de Campinas – SP – Brasil.*

Clarke, G., Wright, J. R. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, pp. 568-581.

Davis, J. S., Kanet, J. J. 1993. Single machine scheduling with early and tardy completion costs. *Naval Research Logistics*, 40, pp. 85-101.

Du, J., Leung, J. Y. T. 1990. Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research*, 15, pp. 483-495.

Eom, D. H., Shin, H. J., Kwun, I. H., Shim J. K., Kim, S. S. 2002. Scheduling jobs on parallel machines with sequence-dependent family set-up times. *International Journal of Advanced Manufacturing Technology*, 19, pp. 926-932.

Feldman, M., Biskup, D. 2003. Single machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering*, 44, pp. 307-323.

Feo, T. A., Resende, M. G. C. 1995. Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, pp. 109-133.

Fernandes, E. R., Ribeiro, C. C. 2005. A multistart constructive heuristic for sequencing by hybridization using adaptive memory. *Electronic Notes in Discrete Mathematics*, 19 pp. 41-47.

Festa, P., Resende, M. G. C. 2004. An annotated bibliography of GRASP. *European Journal of Operational Research*, submitted.

Fleurent, C., Glover, F. 1999. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11, pp. 198-204.

Fowler, J. W., Horng, S. M., Cochran, J. K. 2003. A Hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. *International Journal of Industrial Engineering – Theory applications and Practice*, 10/3, pp. 232-243.

França, P. M., Gendreau, M., Laporte, G., Müller, F. M. 1996. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43, pp. 79-89.

Garey, M., Tarjan, R., Wilfong, G. 1988. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13, pp. 330-348.

Gendreau, M., Hertz, A., Laporte, G. 1992. New insertion and post-optimization procedures for the traveling salesman problem. *Operations Research*, 40, pp. 1086-1094.

Gendreau, M., Laporte, G., Guimarães, E. M. 2001. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133, pp. 183-189.

Glover, F. 1986. Future paths for Integer Programming and links to Artificial Intelligence. *Computers & Operations Research*, 13, pp. 533-549.

Glover, F. 1989. Tabu Search - Part I. *ORSA Journal on Computing*, 1, pp. 190-206.

Glover, F. 1990. Tabu Search - Part II. *ORSA Journal on Computing*, 2, pp. 4-32.

Glover, F. 1996. Tabu search and adaptive memory programming – Advances, applications and challenges. *In: Barr RS, Helgason RV, Kennington JL (Eds), Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research. Kluwer*, 1996, pp. 1-75.

Glover, F., 1998. A template for scatter search and path relinking in Artificial Evolution. *In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (Eds), Lecture Notes in Computer Science*, Springer, 1363, pp. 13-54.

Glover, F., Laguna, M. 1997. Tabu Search. *Kluwer*.

Golden, B. L., Stewart, W. R. (1985). Empirical analysis of heuristics. *In: The traveling salesman problem – A guided tour of combinatorial optimization. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B. (Eds), John Wiley & Sons.*

Gordon V., Proth J. M., Chu C., 2002. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139, pp. 1–25.

Guinet, A. 1991. Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society*, 42/8, pp. 655-671.

Guinet, A. 1993. Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, 31/7, pp. 1579-1594.

Guinet, A. 1995. Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, 6, pp. 95-103.

Hassin, R., Shani, M. 2005. Machine scheduling with earliness, tardiness and non-execution penalties. *Computers and Operations Research*, 32/3, pp. 683-705.

Heady, R. B., Zhu, Z. 1998. Minimizing the sum of job earliness and tardiness in a multimachine system. *International Journal of Production Research*, 36/6, pp. 1619-1632.

Hino, C. M., Ronconi, D. P., Mendes, A. B. 2005. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, 160/1, pp. 190-201.

Hiraishi, K., Levner, E., Vlach, M. 2002. Scheduling of parallel identical machines to maximize the weighted number os just-in-time jobs. *Computers & Operations Research*, 29, pp. 841-848.

Homberger, J., Gehring, H. 2005. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162, pp. 220-238.

Jaines, E.T. 2003. Probability Theory. *Cambridge University Press: United Kingdom*

James, R. J. W., Buchanan, J. T. 1997. A neighbourhood scheme with a compressed solution space for the early/tardy scheduling problem. *European Journal of Operational Research*, 102, pp. 513-527.

James, R. J. W., Buchanan, J. T. 1998. Performance enhancements to tabu search for the early/tardy scheduling problem. *European Journal of Operational Research*, 106, pp. 254-265.

Kanet, J. J., Sridharan, V. 2000. Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, 48/1, pp. 99-110.

Kanet, J. K. 1981. Minimizing the average deviation of job completion times about a common due date. *Naval Research L ogistics Quarterly*, 28, pp. 643- 651.

Kawaguchi, T., Kyan, S. 1998. Deterministic scheduling in computer systems: a survey. *Journal Of Operations Research Society of Japan*, 31, pp. 190-216.

Kim, C. O., Shin, H. J. 2003. Scheduling jobs on parallel machines: a restricted tabu search approach. *International Journal of Advanced Manufacturing Technology*, 22, pp. 278-287.

Kim, D. W., Na, D. G., Chen, F. F. 2003. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer Integrated Manufacturing*, 19, pp. 173-181.

Kim, S. S., Shin, H. J., Eom, D. H., Kim, C. O. 2003. A due date density-based categorising heuristic for parallel machines scheduling. *International Journal of Advanced Manufacturing Technology*, 22, pp. 753-760.

Kolahan, F., Liang, M. 1998. An adaptive TS approach to JIT sequencing with variable processing times and sequence-dependent setups. *European Journal of Operational Research*, 109/1, pp.142-159.

Koulamas, C. 1994. The total tardiness problem: Review and extensions. *Operations Research*, 42, pp. 1025-1041.

Kurz, M. E., Askin, R. G. 2001. Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research*, 39/16 pp. 3737-3769.

Laguna, M., Kelly, J. P., Gonzáles-Velarde, J. L., Glover, F. 1995. Tabu search for the multilevel generalized assignment problem. *European Journal of Operations Research*, 82, pp. 176-189.

Laguna, M., Martí, R. 1999. GRASP with path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11, pp. 44-52.

Lakshminarayan, S., Lakshmanan, R., Papinou, R., Rochette, R. 1978. Optimum single-machine scheduling with earliness and tardiness penalties. *Operations Research*, 26, pp. 1079-1082.

Lee, C. Y., Choi, J. Y. 1995. A genetic algorithm for job sequence problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research*, 22(8), pp. 857-869.

Lee, Y. H., Bhaskaran, K., Pindedo, M. 1997. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, 29, pp. 45-52.

Lee, Y. H., Pindedo M. 1997. Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100, pp. 464-474.

Liao, C. J., Cheng, C. C. 2007. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers and Industrial Engineering*, In press.

Liaw, C-F. 1999. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research*, 26, pp. 679–693.

Logendran, R., McDonell, B., Smucker, B. (In press). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers & Operations Research*.

Martí, R., Laguna, M., Glover, F. 2006. Principles of scatter search. *European Journal of Operational Research*, 169, pp. 359-372.

Mazzini, R. 1998. Estudo de meta-heurísticas populacionais para a programação de máquinas paralelas com tempos de preparação dependentes da seqüência e datas de entrega. *Tese de doutorado, Universidade Estadual de Campinas – SP – Brasil*.

Mazzini, R., Armentano, V. A. 2001. A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operational Research*, 128, pp. 129-146.

Mondal, S. A. 2002. Minimization of squared deviation of completion times about a common due date. *Computers and Operations Research*, 29/14, pp. 2073-2085.

Mondal, S. A., Sen, A. K. 2001. Single machine weighted earliness-tardiness penalty problem with a common due date. *Computers and Operations Research*, 28, pp. 649-669.

Or, I. 1976. Traveling Salesman-Type Combinatorial Problems and their relation to the Logistics of Blood Banking, *Ph.D.Thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL*.

Ovacik, I. M., Uzsoy, R. 1995. Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times. *International Journal of Production Research*, 33, pp. 3173-3192.

Ow, P. S., Morton, T. E. 1989. The single machine early/tardy problem. *Management Science*, 35, pp. 177–191.

Panwalkar, S. S., Dudek, R. A., Smith, M. L. 1973. Sequencing research and the industrial scheduling problem. In: Elmaghraby SE (Eds), *Symposium on the Theory of Scheduling and its Applications*. Springer: Berlin, pp. 29-38.

Panwalkar, S. S., Liman, S. D. 2002. Single operation earliness-tardiness scheduling with machine activation costs. *IIE Transactions*, 34/5, pp. 509-513.

Park, Y., Kim, S., Lee, Y. H. 2000. Scheduling jobs on parallel machines applying neural network and heuristics rules. *Computers & Industrial Engineering*, 38 pp. 189-202.

Parker, R. G., Deane, R. H., Holmes, R. A. 1977. On the use of vehicle routing algorithm for the parallel processor problems with sequence dependent changeover costs. *AIIE Transactions*, 9, pp. 155-160.

Potvin, J. Y., Rousseau, J. M. 1995. An exchange heuristic for routing problems with time windows. *Journal of the Operations Research Society*, 46, pp. 1433-1446.

Potvin, J. Y., Kervahut, T., Garcia, B. L., Rousseau, J. M. 1996. The vehicle routing problem with time windows. Part I: Tabu search. *INFORMS Journal on Computing*, 8/2, pp.158-172.

Rabadi, G., Mollaghasemi, M., Anagnostopoulos, G. C. 2004. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers and Operations Research*, 31, pp. 1727-1751.

Radhakrishnan, S., Ventura, J. A. 2000. Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. International Journal of Production Research 2000, 38, pp. 2233-2252.

Resende, M. G. C., Ribeiro, C. C. 2003. Greedy randomized adaptive search procedures. *In: Glover F, Kochenberger G (Eds), Handbook of Metaheuristics. Kluwer*, pp. 219-249.

Resende, M. G. C., Ribeiro, C. C. 2005. Grasp with path-relinking: Recent advances and applications. *in Ibaraki, T., Nonobe, K., and Yagiura, M. (editors). Metaheuristics: Progress as real problem solvers, Kluwer*.

Rochat, Y., Taillard, E. D. 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics, 1, pp. 147-167.

Rojanasoonthon, S., Bard, J. 2005. A GRASP for parallel machine scheduling with time windows. *INFORMS Journal on Computing*, 17/1, pp. 32-51.

Savelsbergh, M. W. P. 1992. The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal on Computing*, 4 pp. 146-154.

Schaller, J. 2004. Single machine scheduling with early and quadratic tardy penalties. *Computers and Industrial Engineering*, 46/3, pp. 511-532.

Schutten, J. M. J., Leussink, R. A. M. 1996. Parallel machine scheduling with release dates, due dates and family setup times. *International Journal of Production Economics*, 46-47, pp. 119-125.

Sen, T., Sulek, J. M., Dileepan, P. 2003. Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey, *Int. J. Prod. Economics*, 83, pp. 1-12

Serifoglu, F. S., Ulusoy, G. 1999. Parallel machine scheduling with earliness and tardiness penalties. *Computers and Operations Research*, 26, pp. 773-787.

Shannon, C.E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27, pp.379-423, 623-656.

Reprinted with corrections from: cm.belllabs.com/cm/ms/what/shannonday/paper.html

Sidney, J. 1977. Optimum single-machine scheduling with earliness and tardiness penalties. *Operations Research*, 25, pp. 62-69.

Sourd, F., Kedad-Sidhoum, S. 2003. The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 6/6, pp. 533-549.

Sun, H., Wang, G. 2003. Parallel machine earliness and tardiness scheduling with proportional weights. *Computers and Operations Research*, 30, pp. 801-808.

Suriyaarachchi, R. H., Wirth, A. 2004. Earliness/tardiness scheduling with a common due date and family setups. *Computers and Industrial Engineering*, 47/2-3, pp. 275-288.

Szwarc, W., Mukhopadhyay, S. K. 1995. Optimal timming schedules in earliness-tardiness single machine sequencing. *Naval Research Logistics*, 42(7), pp. 1109-1114.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J. Y. 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31/2

Tsai, T.I. 2007. A genetic algorithm for solving the single machine earliness/tardiness problem with distinct due dates and ready times. International Journal of Advanced Manufactoring Technology, 31, pp. 994-1000.

Valente, J. M. S., Alves, R. A. F. S. 2005. Improved heuristics for the early/tardy scheduling problem with no idle time. *Computers and Operations Research*, 32/3, pp. 557-569.

Ventura, J. A., Kim, D. 2003. Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. *Computers and Operations Research*, 30, pp. 1945-1958.

Ventura, J. A., Kim, D., Garriga, F. 2002. Single machine earliness-tardiness scheduling with resource-dependent release dates. *European Journal of Operational Research*, 142, pp. 52-69.

Ventura, J. A., Radhakrishnan, S. 2003. Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research*, 144, pp. 598-612.

Wan, G., Yen, B. P. C. 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. European Journal of Operational Research, 142, pp. 271-281.

Weng, M. X., Lu, J., Ren, H. 2001. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70, pp. 215-226.

Yamashita, D. S., Armentano, V. A., Laguna, M. 2006. Scatter search for project scheduling with resource availability cost. *European Journal of Operational Research*, 169, pp. 623-637.

Zhu, Z., Heady, R. B. 2000. Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers and Industrial Engineering* 38, pp. 297–305.

Apêndice I

Dimensão do espaço de soluções do problema de programação da produção de n tarefas em m máquinas paralelas

Seja n_a o número de tarefas processadas na máquina a, n_b o número de tarefas processadas na máquina b e assim por diante, tal que $n_a + n_b + n_c + \cdots + n_m = n$.

O número de soluções distintas para a máquina a é dado pelo arranjo das n tarefas tomadas em conjuntos de cardinalidade n_a, isto é, $n!/(n-n_a)!$. Para a máquina b, o número de soluções distintas é dado pelo arranjo das $n-n_a$ tarefas restantes, tomadas em conjuntos de cardinalidade n_b, ou seja: $(n-n_a)!/(n-n_a-n_b)!$. Da mesma forma, para a máquina m, o número de soluções distintas é dado por: $(n-n_a-n_b-\cdots-n_{m-1})!/0!$.

O número total de soluções distintas, considerando-se máquinas não-relacionadas e cardinalidade fixa em cada uma das m máquinas é, então, o produto das possíveis soluções em cada máquina, isto é: $(n!/(n-n_a)!)\times((n-n_a)!/(n-n_a-n_b)!)\times\cdots\times((n-n_a-n_b-\cdots-n_{m-1})!/0!)$.

Fazendo-se as simplificações, tem-se que o número total de soluções distintas para cardinalidade fixa em cada máquina é $n!$.

Por outro lado, se as máquinas forem idênticas, o número total de soluções distintas, para cardinalidades fixas em cada uma das m máquinas será $n!/m!$.

Para saber a dimensão do espaço de soluções do problema, é preciso definir todas as possíveis formas de atribuir diferentes quantidades de tarefas a cada uma das máquinas. Para

tanto, considere a tabela a seguir. Na primeira coluna tem-se o total de tarefas atribuídas à primeira máquina. Na segunda tem-se o total de tarefas atribuídas às máquinas restantes. Nas demais colunas tem-se o número de diferentes formas de atribuir as tarefas restantes, quando se dispõe de diferentes números de máquinas além da primeira. A última coluna apresenta a generalização para *m-1* máquinas além da primeira.

Primeira máquina	Demais máquinas					
Tarefas atribuídas	Tarefas atribuídas	1	2	3	4	m-1
		Diferentes possibilidades de atribuição das tarefas restantes				
n	0	1	1	1	1	$(m-2)!/0!/(m-2)!$
n - 1	1	1	2	3	4	$(m-1)!/1!/(m-2)!$
n - 2	2	1	3	6	10	$(m+0)!/2!/(m-2)!$
n - 3	3	1	4	10	20	$(m+1)!/3!/(m-2)!$
n - 4	4	1	5	15	35	$(m+2)!/4!/(m-2)!$
n - 5	5	1	6	21	56	...
...	...	...	...	...	...	...
0	n	1	...	...	...	$(m+m-2)!/n!/(m-2)!$

A soma de todos os termos de qualquer coluna, a partir da terceira, é dada pela expressão $(m-2)!/0!/(m-2)!+(m-1)!/1!/(m-2)!+\cdots+(n+m-2)!/n!/(m-2)!$. Esta soma respeita as condições para aplicação do teorema das diagonais do triângulo de Pascal e o seu resultado é $(n+m-1)!/n!/(m-1)!$.

Portanto, multiplicando este resultado por $n!$, que representa o número de soluções para máquinas não-relacionadas e cardinalidades fixas em cada máquina, tem-se a dimensão do espaço de soluções do problema de programação da produção em máquinas paralelas não-relacionadas, que é $(n+m-1)!/(m-1)!$.

Para máquinas idênticas, a dimensão do espaço de soluções é dada pelo produto do número total de soluções distintas, para cardinalidades fixas em cada uma das *m* máquinas, $n!/m!$, por $(n+m-1)!/n!/(m-1)!$. O resultado é $(n+m-1)!/m!/(m-1)!$.